MOMENTARY ASSESSMENT SYSTEM FOR MOBILE DEVICES

_____

A Thesis

Presented to the

Faculty of

San Diego State University

_____

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

_____

by

Supreeth Shanthakumar

Fall 2011

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Supreeth Shanthakumar:

Momentary Assessment System for Mobile Devices

Yusuf Ozturk, Chair
Department of Electrical and Computer Engineering

Roger Whitney
Department of Computer Science

Alan Riggins
Department of Computer Science

8/16/2011

Approval Date

Copyright © 2011
by
Supreeth Shanthakumar
All Rights Reserved

DEDICATION

In the loving memory of both
my mom and dad.

ABSTRACT OF THE THESIS

Momentary Assessment System for Mobile Devices
by
Supreeth Shanthakumar
Master of Science in Computer Science
San Diego State University, 2011

The presence of the internet and World Wide Web in the contemporary world is undeniable. This nature of the technology has helped in the growth of E-Commerce which has penetrated into most or all types of businesses. The World Wide Web also acts as a huge platform for marketing these businesses and wide range of services. The advent of mobile devices has only helped in the evolution of E-Commerce, helping to reach out to a wider audience.

At the heart of the success of any product lies its ability to listen closely to their customer's needs and complaints. In other words, collection and analysis of valuable data from users or customers is the key. This need also poses a problem of how to accomplish the data collection task without having to lose the essence of the users' feedback or to gather the most accurate data possible.

This thesis project designs a new system which helps gather and aggregate sensitive data from users, to help users have a better understanding of the data. It takes advantage of the ubiquitous nature, location awareness of mobile devices and prevalent web technologies to collect and analyze contextual data from the user. The system allows the creation of survey forms which can be tagged to desired locations in the geographic space. The information gathering process is triggered when a mobile user, who has registered to the system, enters any of these defined locations. The process involves having these mobile users take part in real-time surveys tagged to locations that the user enters. This crowd-sourced data can later be used to analyze, assess an event or gather feedback about the place or location in question.

TABLE OF CONTENTS

PAGE

ABSTRACT v
LIST OF FIGURES ix
ACKNOWLEDGEMENTS xii
CHAPTER
1 INTRODUCTION 1
2 REQUIREMENTS FOR MOBILE ASSESSMENT SYSTEM 6
2.1 Momentary Assessment 6
2.2 Role of Mobile Technology 6
2.3 Growth of Mobile Data Traffic 7
2.4 Location Based Services 8
3 ARCHITECTURE DESIGN 10
3.1 Introduction 10
3.1.1 Presentation Tier 10
3.1.2 Business Tier 11
3.1.3 Data Tier 11
3.2 Design Considerations 11
3.2.1 Google App Engine Java 11
3.2.2 Google Android 12
3.2.3 Adobe Flex 13
3.2.4 OpenDataKit (ODK) 13
3.2.5 Push vs Poll Methodologies 15
3.2.5.1 Polling 15
3.2.5.2 Pushing 16
3.2.6 Android Cloud to Device Messaging Framework (C2DM) 16
3.2.7 Data Exchange Format 17
4 IMPLEMENTATION 20
4.1 Overview 20

4.2 Registration Module....20
4.3 Form Builder Module....21
4.3.1 Question Attributes....25
4.3.2 Question Types....26
4.3.2.1 String Type....26
4.3.2.2 Single Select Type....26
4.3.2.3 Multi Select Type....27
4.3.2.4 Integer Type....27
4.3.2.5 Geolocation Type....27
4.3.2.6 Image Type....29
4.3.2.7 Audio Type....29
4.3.2.8 Video Type....29
4.3.2.9 Image Tagged to Questions....31
4.3.2.10 Video Tagged to Questions....31
4.4 Location Builder Module....32
4.5 Form Submission Viewer Module....33
4.5.1 String Representation....35
4.5.2 Graphical Representation....35
4.5.3 Map Representation....37
4.5.4 Image Viewer Representation....38
4.5.5 Media Player Representation....38
4.6 Data Change Notification Module....39
4.6.1 What is a Task Queue API....39
4.6.2. Structure of C2DM Message....40
4.7 Submission Data Export Module....43
4.8 Admin Module....43
4.9 Location Service Module....45
4.10 Location Alert Module....45
4.11 Survey Retrieval Module....51
4.12 Survey Submission Module....52
4.13 Unregister Module....54
5 USABLE APPLICATIONS....58

5.1 Scenario One 58
5.2 Scenario Two 59
5.3 Scenario Three 60
6 DEPLOYMENT GUIDE 63
6.1 Setting Up Development Environment 63
6.2 Deploying the Application 64
6.2.1 Web Application 64
6.2.2 Mobile Client Application 69
7 USER MANUAL 71
8 FUTURE ENCHANCEMENTS 79
9 CONCLUSION 80
REFERENCES 81
APPENDIX
A ABBREVIATIONS 83
B USEFUL LINKS 85

LIST OF FIGURES

PAGE

Figure 1.1. Momentary assessment system use case diagram. 5
Figure 2.1. Cisco mobile data traffic forecast per month by 2015. 8
Figure 3.1. 3-tier architecture of momentary assessment system. 10
Figure 3.2. C2DM framework explained. 18
Figure 3.3. Data exchange format sample. 18
Figure 4.1. User registration activity diagram. 21
Figure 4.2. Sequence diagram explaining user registration process. 22
Figure 4.3. Activity diagram for form builder module. 23
Figure 4.4. Sequence diagram showing the process of saving a form. 24
Figure 4.5. Sample form definition. 25
Figure 4.6 String question type. 26
Figure 4.7. Single select question type. 27
Figure 4.8. Multi select type question. 28
Figure 4.9. Integer type question. 28
Figure 4.10. Geolocation type question. 29
Figure 4.11. Image type question. 30
Figure 4.12. Audio type question. 30
Figure 4.13. Video type question. 31
Figure 4.14. Image tagged to question. 32
Figure 4.15. Video tagged to question. 32
Figure 4.16. Database mapping for location and form module. 33
Figure 4.17. Activity diagram for submission viewer module. 34
Figure 4.18. Sequence diagram for submission viewer process. 35
Figure 4.19. Class diagram for SubmissionXMLTable. 36
Figure 4.20. String representation for submission. 36
Figure 4.21. Graphical representation for submission. 37
Figure 4.22. Map representation for submission. 38

Figure 4.23. Image viewer representation for submission. 38
Figure 4.24. Media player representation for submission. 39
Figure 4.25. Data change notification activity diagram. 40
Figure 4.26. Sequence diagram for data change notification. 41
Figure 4.27. Structure of queue.xml. 41
Figure 4.28. Structure of sample C2DM message. 42
Figure 4.29. Activity diagram for survey export module. 43
Figure 4.30. Sequence diagram for exporting submission data. 44
Figure 4.31. Sample Google app engine user service code. 44
Figure 4.32. Class diagram for LocationService. 46
Figure 4.33. Class diagram for C2DM message. 47
Figure 4.34. Sequence diagram explaing the location service module. 48
Figure 4.35. State diagram for location alert module. 48
Figure 4.36. Sequence diagram explaining the location alert process. 50
Figure 4.37. Activity diagram for survey retrieval module. 52
Figure 4.38. Sequence diagram explaining the survey retrieval process. 53
Figure 4.39. Class diagram for download tasks. 54
Figure 4.40. Sequence diagram explaining the survey submission process. 55
Figure 4.41. Survey uploading notification on mobile device. 56
Figure 4.42. Sequence diagram explaining the user UnRegistration process. 57
Figure 5.1. Application – Survey form to analyze employee effectiveness. 59
Figure 5.2. Application – Survey form to create Yelp like review system. 61
Figure 5.3. Application – Survey form to gather data on demand. 62
Figure 6.1. Android C2DM signup application information. 64
Figure 6.2. Android C2DM signup contact information. 65
Figure 6.3. Google app engine new application welcome screen. 65
Figure 6.4. Google app engine create an application. 66
Figure 6.5. Export flex application. 66
Figure 6.6. How to deploy to Google app engine. 68
Figure 6.7. App engine deploy signin. 68
Figure 6.8. App engine project settings. 69
Figure 6.9. Export signed Android application package. 70

Figure 7.1. Provide admin access to user.....71
Figure 7.2. Admin panel to provide system access to users. 72
Figure 7.3. Form creation/builder panel. 72
Figure 7.4. Location dashboard view.....73
Figure 7.5. Form dashboard showing form details. 74
Figure 7.6. Submission viewer showing question carousel. 75
Figure 7.7. Device monitoring panel. 76
Figure 7.8. Settings to install Android application from unknown sources.....77
Figure 7.9. Android application showing the sequence of the registration process. 77
Figure 7.10. Android application how to disconnect/unregister.....78
Figure B.1. Homepage of my master's thesis project. 86

ACKNOWLEDGEMENTS

No thesis project is a venture undertaken by one person. It is indeed a collaborative effort requiring the support, guidance & advice of many.

I would like to thank my family back home who provided me with an opportunity and encouraged me to purse my Master's degree. Without your unwavering support and help, I could never have made it through this process. To my brother, who always inspired me to dream and encouraged me to follow through with it every time.

Special thanks to my thesis advisor and chair, Prof. Yusuf Ozturk. Thank you for believing in me, your patience, your continual support and encouragement, and also for your incredibly thoughtful guidance and input. It was an honor and a privilege to work with you.

Last but not least, thanks to all of my friends who always gave me a perspective on my project and who made sure that I was always on the right path.

CHAPTER 1

INTRODUCTION

The presence of the internet and the World Wide Web in the contemporary world is undeniable. Today, almost everybody interacts with the web in one form or another. Social networking being the biggest internet phenomenon has become an integral part of many lives, which were targeted especially towards the younger generation. The omnipresence nature of the web has expanded the reach of E-Commerce which has made its way into all types of businesses. Electronic Commerce [1], commonly known as E-Commerce consists of buying and selling of products or services over the internet. Apart from buying and selling of products it also includes the process of developing, marketing, servicing for those products. This has grown at an exponential rate due to the widespread nature of the internet. Supply chain management, Internet marketing, online transaction processing, electronic data interchange, inventory management systems and automated data collection systems are all part of the whole E-commerce ecosystem.

The advent of mobile devices like smartphones, PDA (see Appendix A), tablets, laptops have only helped in the evolution of E-Commerce. A smartphone [2] is a high-end mobile phone which not only has the basic telephony features but also houses rich multimedia capabilities like high-resolution touchscreens, built-in cameras, GPS navigation, Wi-Fi and mobile broadband access. With the birth of these devices, every business has expanded their services to tailor to this growth by building mobile applications and mobile friendly websites. These devices provide a ground for huge market consumption.

One of the key ingredients for the success of any business or a study is the feedback they get from consumers, users or participants. There is a demand for analytical tools which help these organizations to make their services more tailored to the audience at hand. The more data or information they get back from the users, the better it is for them to assess the situation and view from the consumers perspective what they truly feel or what they truly need. One successful method to accomplish this task is called crowdsourcing. This is a term coined jointly by Jeff Howe in June 2006 issue of *Wired* magazine article “The Rise of

Crowdsourcing" [3], which describes a new model based on web-technologies helping business harnesses the creative solutions of a network of distributed individuals through what amounts to an open call for proposals. Howe defines [3] crowdsourcing as the act of a company or institution outsourcing a function to a network of people in the form of an open invitation, which was once performed by employees. This can not only take the form of peer-production, but also by sole individuals. The notion of open invitation format and large network of potential laborers act as the only prerequisite [3].

During the past decade crowdsourcing has grown from a number of initiatives like Wikipedia and Mechanical Turk [4], to a technique that today attracts and engages millions of people worldwide. Crowdsourcing has taken new forms as the computing systems are becoming an intimate part of physical and social contexts. Increasingly, crowds are engaged through their mobile devices to capture, share and validate huge amounts of data e.g. reporting security threats or capturing social events [4]. An example of how crowdsourcing has been put to use in real world applications, can be seen in Google Maps for mobile devices. When the user requests driving directions, their complex algorithm formulates the best, fastest route from the source destination to the target destination. This is accomplished when the phone, which is equipped with GPS, sends anonymous bits of data back to Google describing how fast the user is moving in the traffic [5]. When these large sets of data are combined with the speed of other devices on the road, across thousands of phones moving around a city at any given time, they can formulate a good picture of live traffic conditions. They continuously combine this data and sent it back to the users for free in Google Maps traffic layer. The best part of this approach is that, it almost requires zero effort from the users end, expect to agree to their terms and conditions which give them the right to send anonymous data back to the server. The more people that participate, the better and more accurate are the resulting traffic reports.

The rise of Web 2.0 paradigm has given way to the rise of the global phenomenon called Social Networking. A social networking service [6] is a web-based service, platform that focuses on building social networks or social relations among people who, for example share interests and/or activities. One of the most famous examples of social networking service, which has become a household name, is Facebook. It is a web-portal where people create profiles of themselves, connect with friends and families, share photos, thoughts,

create events and send invites to those who are connected to them, etc. Due to sheer amount of people who are actively participating in these services on a daily basis, businesses find this as the best medium to advertise their services and/or products.

Another example of a social networking service which is focused mainly on acting as a platform for businesses to gather feedback from their customers is Yelp. It is an online city guide that helps people find good places to eat, shop, drink, play, based on opinions, reviews, and ratings by an active local community [7]. But there are some minor setbacks and/or questions to this way of promoting business, like how accurate these reviews actually are. Also there arises a question of consist these reviews are within the context space.

What is context? “Context is any information that can be used to characterize the situation of an entity, where an entity can be a person, place, physical or computational object” [8]. There are different types of context as defined by Göker and Myrhaug [9].

- **Spatio-temporal context**
  What is the physical location of the user? What is the time of the day? Location has been frequently been used in context aware applications as it is one of the most useful context information.
- **Social context**
  The user’s friends and relatives, the user’s social position.
- **Environment context**
  This relates to the information having to do with user surroundings and environment in a very general sense.
- **Personal context**
  What the user is interested in, is the user suffering from any ailment, does the user have allergies?
- **Task context**
  What is the user currently doing or will be doing in the future.

A service that uses context information as an input and adjusts its behavior accordingly is a context aware service. We can give examples of such services:

- If we use a device to detect presence in a room, to automatically regulate the temperature, saving energy for heating and cooling if the room is unoccupied.
- The network connection on a mobile device automatically changes according to the user’s preferences and location: for example, when a user is at work, he can switch to wireless LAN.
- During a meeting, all but emergency incoming calls are redirected to the user’s voice mail.

This study aims at collecting contextually rich momentary data from the user, focusing on the user's current physical location, which can be later used for analysis.

What is momentary data? It is considered as the data or the information that the subject can provide while they are still actively engaged in a task or at the time when the target behavior has occurred – in the moment. This data can be more accurate and helpful than the information which can be provided after a certain period of time since the occurrence of the event or target behavior. While the user is still engaged, he can provide a more detailed description of his mood.

The Figure 1.1 describes the process employed by the momentary assessment system. It consists of a form builder module which helps the users with admin privileges to create survey questionnaires and tag them to particular locations which also can be created using the system. The questionnaires can request multiple types of input from the users, like a question can request a text, number only entry or even ask the user to capture an image, video or an audio message. When the mobile users register to this service, they agree to be part of the survey when they find themselves in those already chosen points of interest. When the mobile user enters these zones of interest, which is detected by the built-in GPS on the mobile device, he is alerted to act as a participant in the study or survey by answering the questionnaire. After the user has completed the survey he saves the information and uploads the data to the server where this data is presented to the owner of the questionnaire to help aid in his study, hence acting as an analytical tool.

As previously mentioned, businesses need feedback from their users and this tool can help them in this cause. Example, a store owner can tailor the system to his business, create surveys targeted to gather information regarding the user's level of satisfaction with the service and tag them to each of his store. When a customer or a registered mobile user physically enters these stores, he can take part in this survey and send his sensitive data anonymously back to the server. Using this data the owner can assess how efficiently his staff is serving the needs of the customer. This is just one example of how this system can be put to use in real life scenarios, few more examples are illustrated in depth in Chapter 5: Usable Applications.

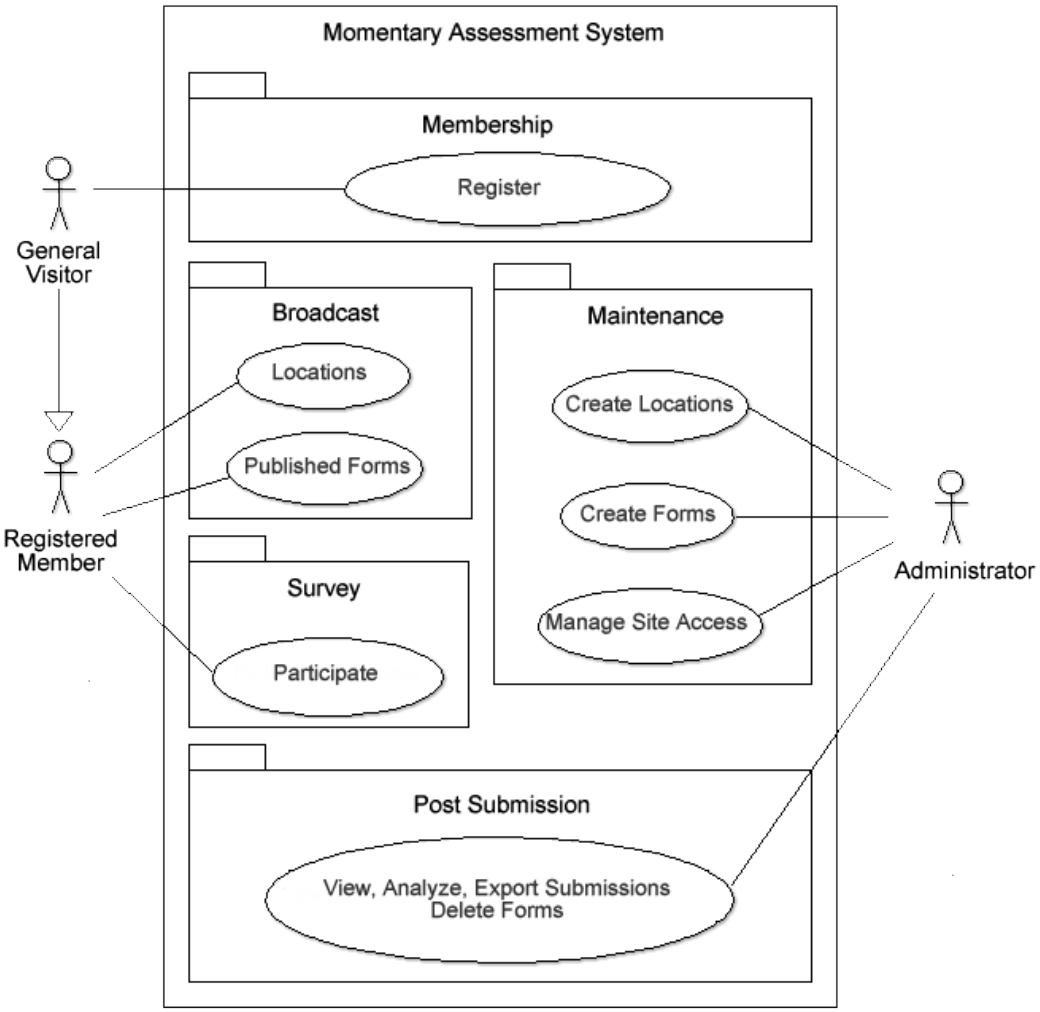


Figure 1.1. Momentary assessment system use case diagram.

CHAPTER 2

REQUIREMENTS FOR MOBILE ASSESSMENT SYSTEM

2.1 MOMENTARY ASSESSMENT

One of the common techniques for gathering data is paper-and-pencil self-monitoring (PM) technique [10], which was used to gather information on behaviors related to drinking and other internal states like mood. In this method, participants would enter information on cards or books whenever the target behavior occurred. Even though this method has the limitation of relying on self-report, it proved as effective, or more, as other traditional methods of measuring a situation. But PM had some limitations that were not easily addressed, despite its advantages which included possibility of faked data, or deficient data and inability to provide base-rate data which is important in establishing associations between potential antecedents like mood and the target behavior. Ecological momentary assessment (EMA) [11] was one of the most sophisticated alternative to PM, in which participants interacted with small, hand-held computers to initiate entries of the occurrence of a target behavior or event driven by location and respond to random prompts that provide base-rate. EMA techniques had advantages like data are collected in real time, it could not be faked since each entry is tagged with a date, time and participants could not skip items and must complete a standard interview which provides quality control of data.

2.2 ROLE OF MOBILE TECHNOLOGY

In today's world, mobile devices act as the predominant means of communication. The mobile telecommunication system transformed completely into 3G or 3rd generation mobile telecommunications during the latter half of the past decade. This meant that the mobile users not only could transmit voice over the communication lines but also data, eventually giving birth to a whole suite of data-rich services. These services range from connecting with other users for example via email, Instant messaging to a fully implemented mobile web browser, enabling users to get access to the internet which was previously only possible through desktop computers.

Ever since the advent of smartphones, the number of mobile users have been growing at a great pace throughout the globe. This growth and the personal nature of mobile devices provide businesses, organizations a vast opportunity to establish their market into the mobile space. There are already many services in practice which take advantage of the mobile nature, like

- Coupon services - which can be shown on the mobile device at a local business to redeem discounts
- Check-in services – which allows a mobile user to check-in on a mobile app, broadcasting his location to the entire world, and also allows him to tag messages/tips to the checked-in business to help other people choose if the place is right for them.
- Navigation services – which help users, navigate around a city using the built-in GPS system.
- Mobile advertisements

2.3 GROWTH OF MOBILE DATA TRAFFIC

There are number of studies going on to predict the growth of mobile usage and data traffic. Cisco Visual Networking Index (VNI) Forecast is an ongoing initiative to track and forecast the impact of visual networking applications on global networks. One of their papers presented the growth trends and global mobile data traffic projections.

These studies revealed that global mobile data traffic grew 2.6-fold in the year 2010, which nearly tripled for 3 years in a row [12]. The average mobile network downstream speed in 2009 was 101 kilobits per second (kbps) and in 2010 it was 215 kbps [12]. The average mobile network connection speed for smartphones came up from 625 kbps in 2009 to 1040 kbps in 2010. The same study reported that average smartphone usage doubled in 2010 [12].

According to their research and study, overall data traffic is expected to grow to 6.3 exabytes per month by 2015, which is a 26-fold increase over the year 2010 [12] as shown in Figure 2.1.

The following are a few factors in the growth of mobile data on average usage [12]:

- Mobile network connection speeds are going to increase, in turn the average bit rate of content accessed through the mobile network will also increase. This will lead to the increase in the proportion of streamed content as compared to side-loaded content.
- The increase in battery life of mobile devices will increase mobile minutes usage.

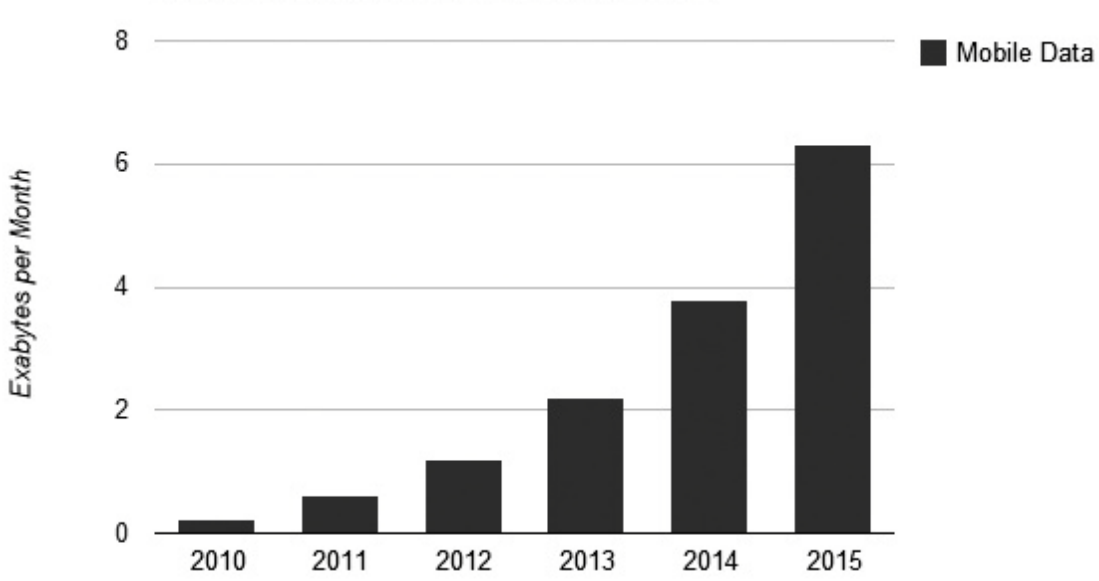


Figure 2.1. Cisco mobile data traffic forecast per month by 2015.

- As mobile network capacity improves, operators are more likely to offer mobile broadband packages comparable in price and speed to those of fixed broadband, thereby encouraging mobile broadband substitution.
- The shift towards unicast from broadcast video will affect mobile networks as much as it will affect fixed networks. In unicast there is one data stream per user like Internet radio and Internet video. Where as in broadcast there is one data stream which serves many users. This shift means that traffic can increase dramatically even while the total amount of time spent watching video remains relatively constant.

2.4 LOCATION BASED SERVICES

Location sensitive smartphone devices expand the horizon of mobile services and makes way for Location Based Services (LBS). These are services that take advantage of the mobile users' location and exploit any of the several technologies to locate or place the user in a geographic location to help provide relevant services. It could either use Global Positioning System (GPS) or other location tracking and positioning systems.

Location tracking and positioning systems can be categorized by the techniques used to determine mobile device location. Typically, real-time location systems (RTLS) can be grouped into four basic categories of systems [13] that determine position on the basis of the following:

- Cell of origin (nearest cell)
- Distance (lateration)

- Angle (angulation)
- Location patterning (pattern recognition)

With the advancements in technology and growth of mobile market, these services become prevalent which were not possible on such a global scale a decade or so ago. Location Based Advertising is one of the example which deliver context based advertisements. With the emergence of mobile business and localization technologies, a new type of marketing communication is possible: Mobile advertisement with localization of the users.

All of these factors points to the need for a mobile assessment system which can communicate with the participant at any given time, helping to gather sensitive analytical data instead of relying on the age old pencil-and-paper technique. This thesis project follows the EMA technique and showcases a system to collect survey data from mobile users triggered by locations of interest or on demand by the owner of the application.

CHAPTER 3

ARCHITECTURE DESIGN

3.1 INTRODUCTION

The momentary assessment system described in this thesis project uses the traditional 3-Tier architecture. A layered architecture of the system is shown in Figure 3.1. This chapter also briefly presents each system layer and its components.

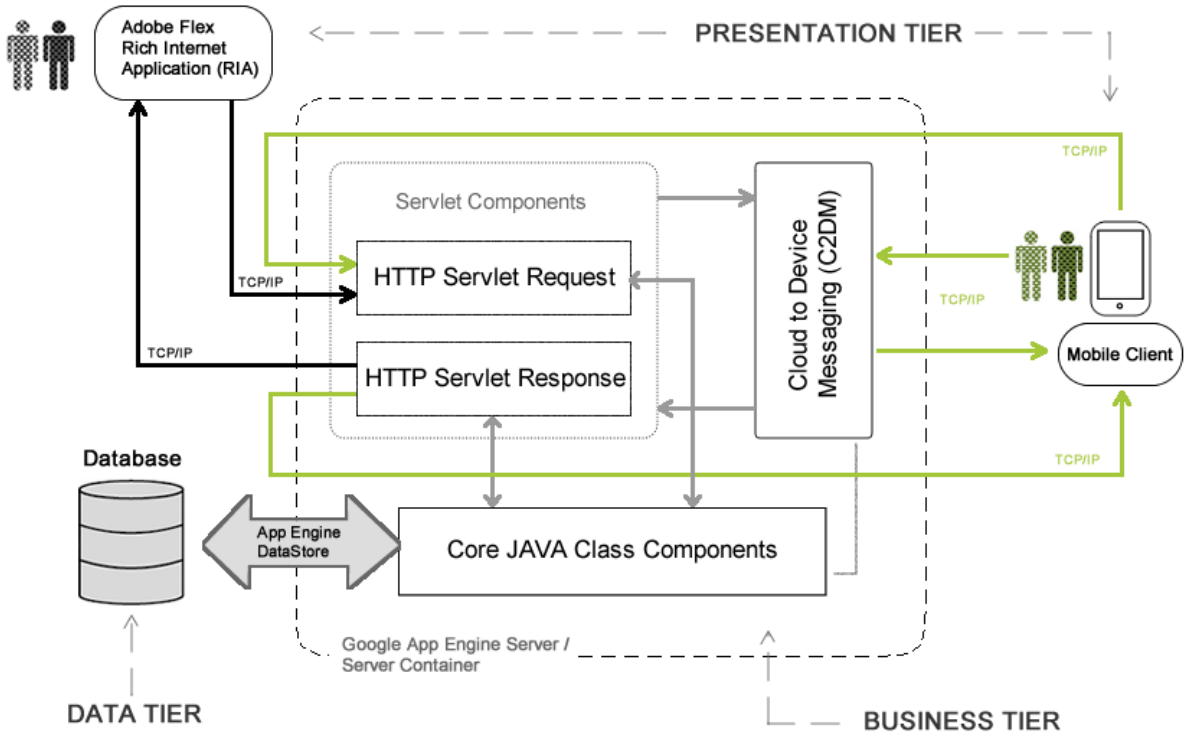


Figure 3.1. 3-tier architecture of momentary assessment system.

3.1.1 Presentation Tier

The presentation layer is the one with which the end users interact with. It consists of Adobe Flex application hosted on the web browser and also the Android application for mobile users.

3.1.2 Business Tier

The middle tier consists of the server side logic which interacts directly with the underlying database. It has the implementation of servlets which are hosted on Google App Engine Java. This layer is responsible for managing the communication between the web browser, Android application and the database present in the Data layer. It also comprises of the C2DM framework which is responsible for keeping the mobile devices updated with the latest data.

3.1.3 Data Tier

This layer manages the storage of the actual data. Since the application is hosted using Google App Engine, it comprises of the App Engine Datastore which is a schemaless object datastore, with a query engine and atomic transactions. The Java SDK includes implementations of Java Data Object (JDO) and Java Persistence API (JPA) interfaces, as well as the low-level datastore API.

3.2 DESIGN CONSIDERATIONS

This section discusses all the standards, technologies and open source projects that were considered to build the momentary assessment system.

3.2.1 Google App Engine Java

The webserver is built using the Google App Engine Java framework with servlets accepting and processing the requests. App Engine helps build web applications using standard Java technologies and run them on Google's scalable infrastructure. The Java environment provides a Java 6 JVM, a Java Servlets interface, and support for standard interfaces to the App Engine scalable datastore and services, such as JDO, JPA, JavaMail, and JCache.

App Engine uses the Java Servlet standard for web applications. We can provide our app's servlet classes, JavaServer Pages (JSPs), static files and data files, along with the deployment descriptor (the web.xml file) and other configuration files, in a standard WAR directory structure. App Engine serves requests by invoking servlets according to the deployment descriptor.

The JVM runs in a secured "sandbox" environment to isolate your application for service and security. The sandbox ensures that apps can only perform actions that do not interfere with the performance and scalability of other apps. For instance, an app cannot spawn threads, write data to the local file system or make arbitrary network connections. An app also cannot use JNI or other native code. The JVM can execute any Java bytecode that operates within the sandbox restrictions.

The App Engine datastore provides robust, scalable storage for your web application, with an emphasis on read and query performance. An application creates *entities*, with data values stored as *properties* of an entity. The app can perform queries over entities. All queries are pre-indexed for fast results over very large data sets.

3.2.2 Google Android

Android is an open source operating system for mobile devices. Android was initially developed by Android Inc., and sold to Google in 2005. On November 2007, the Open Handset Alliance (OHA) was announced amongst a consortium of several top companies. The goal was to develop an open mobile platform every developer to contribute towards improving the performance and features of the product.

Android is built on top of linux kernel and GNU. It is a software stack for mobile devices that includes an operating system, middleware and key applications. Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Developers use the Android SDK to build applications. This SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

3.2.3 Adobe Flex

The web client front end is built using Adobe Flex. Flex is a highly productive, free, open source framework for building intuitive mobile, web, and desktop applications that make it a lot easier for users to view and interpret the data represented. These applications are often called as Rich Internet Applications (RIA). Flex allows you to build web and mobile applications that share a common code base, reducing the time and cost of application creation and longer term maintenance. See Appendix B to get the link to the already deployed project as part of the testing process.

Flex uses ActionScript which is the core language of Flash player based on the ECMAScript standard. The Flex SDK comes with a wide set of user interface components including buttons, list boxes, trees, data grids, several text controls, and various layout containers. Charts and graphs are available as an add-on. Other features like web services, drag and drop, modal dialogs, animation effects, application states, form validation, and other interactions round out the application framework.

In a multitiered model, Flex applications serve as the presentation tier. Unlike page-based HTML applications, Flex applications provide a stateful client where significant changes to the view do not require loading a new page. Similarly, Flex and Flash Player provide many useful ways to send and load data to and from server-side components without requiring the client to reload the view. Though this functionality offered advantages over HTML and JavaScript development in the past, the increased support for XMLHttpRequest in major browsers has made asynchronous data loading a common practice in HTML-based development as well.

The Flex 4 development environment is called Adobe Flash Builder, formerly known as Adobe Flex Builder, which includes the Flex compiler from Adobe. The framework has been designed for continuous collaboration between designers and developers.

3.2.4 OpenDataKit (ODK)

The entire momentary assessment system is built on top of Open Data Kit. It is an open-source suite of tools that helps organizations author, field, and manage mobile data collection solutions. Their goals are to make open-source and standards-based tools which

are easy to try, easy to use, easy to modify and easy to scale. See Appendix B for the link to ODK source code.

ODK's core developers are researchers at the University of Washington's Department of Computer Science and Engineering department and active members of Change, a multi-disciplinary group at UW exploring how technology can improve the lives of under-served populations around the world.ODK began as a google.org sponsored sabbatical project under the direction of Gaetano Borriello in April of 2008 at Google's Seattle offices. It is funded by a Google Focused Research Award and supported by a growing community of developers, implementers and users.

ODK are members of the OpenRosa Consortium and active participants in the JavaRosa project. Before the establishment of OpenROSA consortium, there were a number of free and open source tools available for gathering information/collection of data on mobile devices. This eventually led to a lot of implementations with each having a different approach but it lacked standards and openness. This became the source for the birth of OpenROSA consortium, whose goal is to foster open-source, standards-based tools for mobile data collection, aggregation, analysis and reporting.

This solution conforms to standards based on XForms specification which is the official W3C standard for next-generation data collection and interchange. The openness of this solution and the above said conformance ensured that different projects could easily share code, data, ideas and the infrastructure.

JavaROSA is one of the successful projects that had its source in OpenROSA. JavaROSA is an open-source J2ME codebase that conforms to OpenROSA standards. It is being developed for a wide range of application including disease surveillance, household surveys, and collection of longitudinal data for electronic medical records, guiding health workers through medical protocols at the point of care, and supporting community health workers.

The ODK questionnaires are written in xml format and can be created manually or generated automatically using an online interface. It provides the following modules as part of their open source set of tools.

- Build:
  This is a tool to build the form or survey.

- Collect:
  This uses the Android platform to collect information from the mobile user which supports various methods of input, e.g. video, audio, photo, GPS location.
- Aggregate:
  This is a ready to deploy server repository to manage the data collected, provide interfaces to extract data, e.g. spreadsheets, queries etc. This is implemented on the Google App Engine which frees the users from setting up their own scalable web service.

The other tools in the suite, Manage, Validate, and Voice, were not used as a part of this thesis project.

3.2.5 Push vs Poll Methodologies

The emergence of mobile devices as the leading personal communications devices has attracted organizations and marketers as a platform to deliver their services. Most of the modern services don't restrict data to reside only on the device but instead in the cloud or server. One of the fundamental challenges for these types of mobile applications/services is how to keep the data on the device fresh and up-to-date. There are two basic techniques to accomplish it, namely polling and pushing.

3.2.5.1 POLLING

This is a fairly simple technique to implement. In polling, the application periodically checks to see if any data has changed on the server. It might be a simple approach but a lot of times the data on the server isn't changing as frequently as the application might poll. So it is obvious that this approach could take a toll on the battery life of the device. This is because when we are polling or trying to access data or use data network we are powering up the radio. Using the radio translates to, the device consuming battery power. And even after the data retrieval process is finished the radio stays on for a couple more seconds.

The impact of polling on battery can be explained based on one of the routine measurements taken by the Android team at Google [14]. When the device is idle, it might even have an open TCP connection with the server, but there's no data being transmitted or received. The baseline power drawn is about 5 – 8 milliamps (mA). But when we start reading data that same power usage can jump up to 115mA. When we write, start transmitting data the power usage jumps up to 180-200mA or even more which is costlier

than reading. So for a short poll we might have consumed as much as 2.5 milliamp hours and this could add up. If the application is set to poll every 5 minutes, then that adds up to 144 milliamp hours which is quite a bit of battery usage. A typical battery gives you about 1200 to 1400 milliamp hours, so that's over 10% of the battery, from one application polling every 5 minutes. We can scale the polling to every 15 minutes or so, which would do a lot better of about 48 milliamp hours.

This gives rise to the question as to when we want to poll. There is a tradeoff between freshness of the data and efficiency. If you poll frequently, we will have fresh data but the application is not going to be efficient. If we poll less frequently, the data is not going to be fresh but its better on the battery life. However polling is completely appropriate for some types of content. If we have content that actually changes constantly like stock quotes, news headlines etc and the application is tolerant of showing slightly stale data, you can poll very infrequently and then when the user is actually starting to view that data the application can then refresh the data. But typically we don't want to poll, we want to push to keep the data fresh and to reduce power consumption.

3.2.5.2 PUSHING

Push based systems provide both freshness and efficiency, and they do that by only making use of data when it's useful. In a push system, an application or service will establish and maintain a persistent connection with a server in the cloud. This allows the server to send data to the application only when data has changed.

On a commercial Android device push functionality is already in use for Google contacts, calendar, Gmail sync, Google voice and also Android Market. Implementing push based system on Android is possible since the Android API allows anyone to run a background service to establish and maintain a connection. As of Android 2.2 platform this push based system is made available to developers as Android Cloud to Device Messaging (C2DM).

3.2.6 Android Cloud to Device Messaging Framework (C2DM)

Android Cloud to Device Messaging (C2DM) is a service that helps developers send data from servers to their applications on connected Android devices. The service provides a

simple, lightweight mechanism that servers can use to instruct the relevant mobile applications to contact the server directly, to fetch updated application or any required user data. The C2DM service handles all aspects of queuing of messages and delivery to the target application running on the target device.

This framework allows third-party application servers to send lightweight messages to their Android applications. The messaging service is not designed to send a lot of data via the messages. Rather, it should be used to wake the mobile application and let them know that there is new data available on the server, so that the application can query the server and fetch the relevant data.

An application on an Android device doesn't need to be running to receive messages. The system will wake up the application via Intent broadcast when the message arrives, as long as the application is set up with the proper broadcast receiver and permissions. It requires devices running Android 2.2 or higher that also have the Market application installed. However, you are not limited to deploying your applications through Market. It uses an existing connection for Google services. This requires users to set up their Google account on their mobile devices.

Figure 3.2 [14] shows an Android device already established a connection with Google's connection servers. An application server will send a simple HTTP post to Google's cloud-to-device-messaging front ends. It receives the request and stores the message. Then it routes the request to the appropriate connection server that's housing a connection from the target device, which will in turn go ahead and send the message to the device.

On the device side two things happen, in parallel.

1. The message is converted into an intent which is broadcast to wake up the registered application.
2. The device will acknowledge the receipt of that message, so that it can be cleared from the queue on the Google's datastore which houses all the C2DM messages.

3.2.7 Data Exchange Format

All the data exchange between the server and web-client, server and connected devices is implemented using XML. The backend server consists of servlets waiting for

requests which could either be fetch, update or delete. These servlets respond back using standard XML notation.

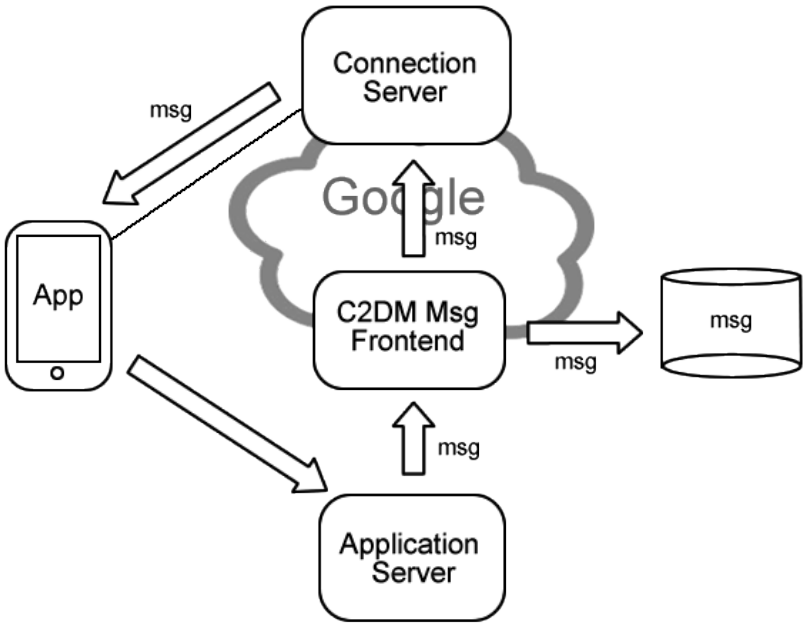


Figure 3.2. C2DM framework explained.

Example: When the mobile device is first connected to the server, i.e. when the user has registered, it will contact the server to fetch all the desired locations which are already present in the system. An example request url is given here: http://momentaryassessment.appspot.com/locationListXML

The above request to the server results in a response as shown in Figure 3.3.

```
<locations>
        <location id="1303178716615" name="Sample Location#1"
lat="32.77384246291378" lng="-117.07454094050595" radius="50"
createdBy="test@example.com"/>
        <location id="1303426809079" name="Sample Location#2"
lat="25.776958836969538" lng="-80.1767166868164" radius="250"
createdBy="test@example.com"/>
</locations>
```

Figure 3.3. Data exchange format sample.

On the mobile client side, this xml is data is parsed and the location details are gathered and saved locally on the device. The above example shows how the exchange of data occurs in the system.

CHAPTER 4

IMPLEMENTATION

4.1 OVERVIEW

This chapter explains all the different modules that comprise the momentary assessment system. Since the application has both the server side and mobile client side, they have been broken up accordingly to give you an in-depth view of the internal workings of the system.

4.2 REGISTRATION MODULE

This module is responsible for successfully registering a mobile device with the server.

The Figure 4.1 depicts the activity diagram of the user registration process. Figure 4.2 shows the sequence diagram of the same process and shows all the classes involved.

When the user installs the mobile client application and runs it for the very first time he is requested to register with the server to establish connection. The pre-requisite for registering is that the user should have at least one Gmail account on his device and has Android Market application installed on the device. Once the user selects the desired gmail account the device is registered on the Google's C2DM server and the *deviceRegistrationID* is forwarded to the system's app server residing on the Google App Engine.

Once the device has been registered, the user information along with the *deviceRegistrationID* is stored on the app server's datastore. Also after the device is registered, the mobile device requests information about all the locations that it might be interested in. Using this registration information, the system can send relevant messages to the mobile device.

Messages are sent to the connected devices by the system when:

1. New location is added to the system
   This message contains information regarding the newly added location, so that the client device can add this location to its local database.

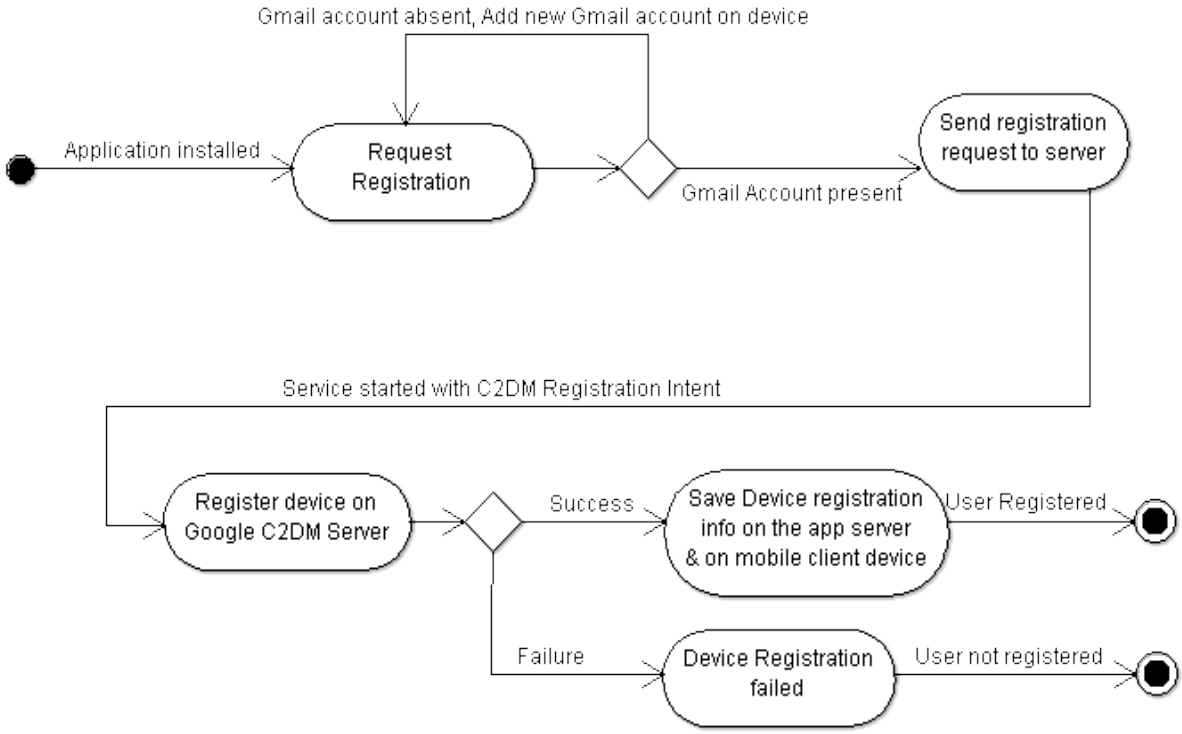

Gmail account absent, Add new Gmail account on device
Application installed
Request Registration
Gmail Account present
Send registration request to server
Service started with C2DM Registration Intent
Register device on Google C2DM Server
Success
Save Device registration info on the app server & on mobile client device
User Registered
Failure
Device Registration failed
User not registered

Figure 4.1. User registration activity diagram.

1. Existing location is edited or deleted
   This message contains the *locationID* of the updated location, which indicates that the client device needs to update or delete the already existing location with newer information.
2. The system explicitly asks all the devices connected to take part in a survey. For example in the event of an emergency or upon the occurrence of an event for which the administrator feels the need to gather information from the user.

   For more information and format of these messages please refer Section 4.6 Data Change Notification Module.

4.3 FORM BUILDER MODULE

As mentioned before the ODK makes use of XForms to define and render a form. The form builder module helps the users to generate a form using an intuitive user interface, which is built on top of ODK's aggregate module. He can then publish these survey forms to the users who are connected to the server.

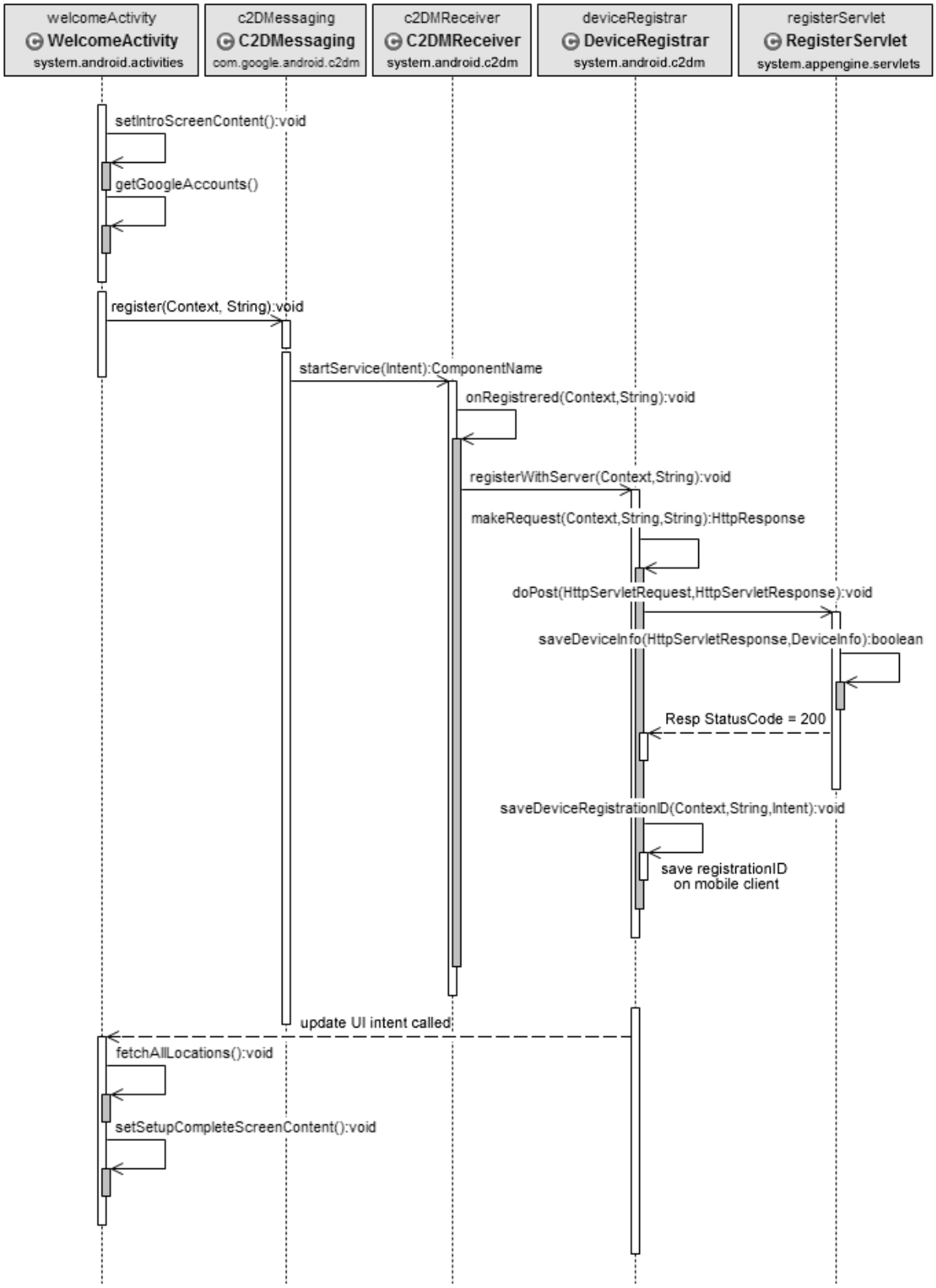


Figure 4.2. Sequence diagram explaining user registration process.

The Figure 4.3 shows the activity diagram for the form building process employed by this module. A form is a survey tool which consists of a set of questions. Each of these forms then can be associated to any location (also defined by the users) available in the system.

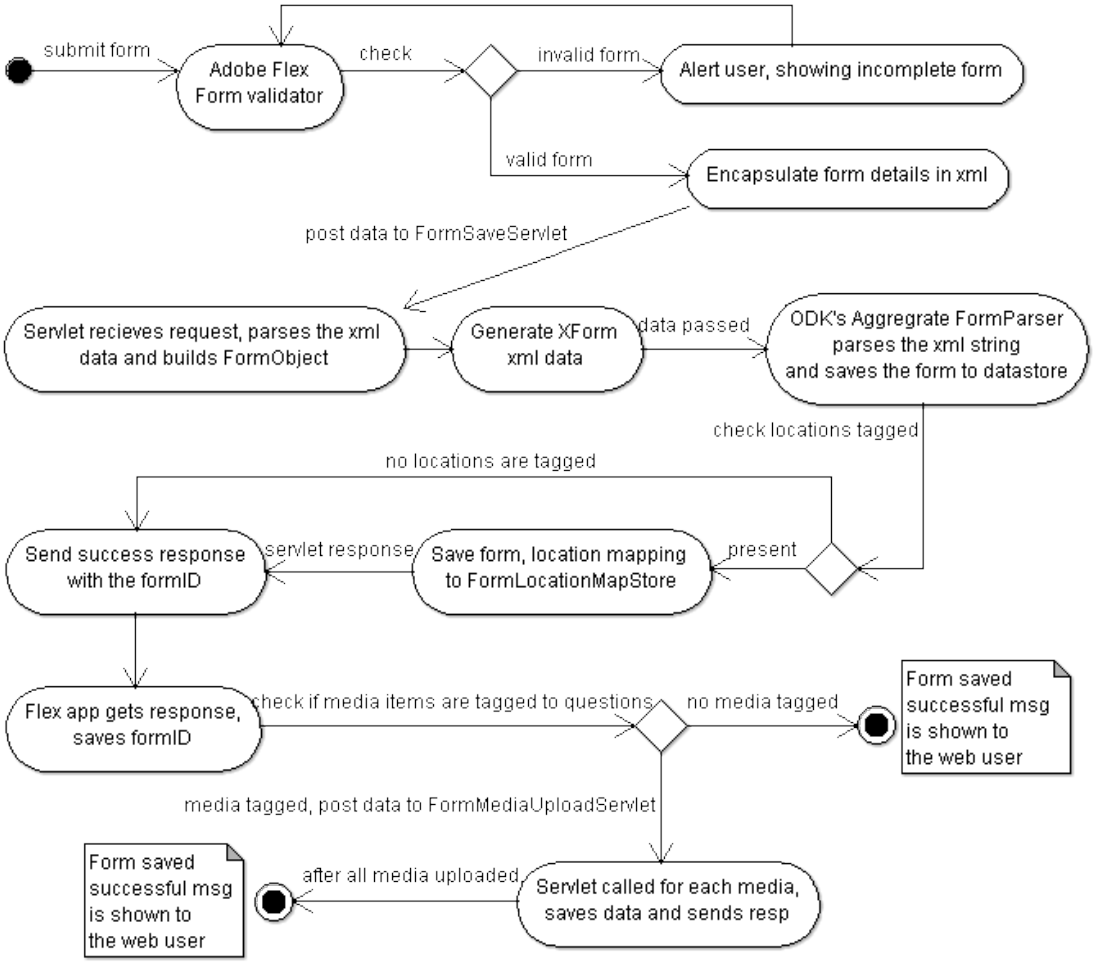

submit form
Adobe Flex Form validator
check
invalid form
Alert user, showing incomplete form
valid form
Encapsulate form details in xml
post data to FormSaveServlet
Servlet recieves request, parses the xml data and builds FormObject
Generate XForm xml data
data passed
ODK's Aggregrate FormParser parses the xml string and saves the form to datastore
check locations tagged
no locations are tagged
Send success response with the formID
servlet response
Save form, location mapping to FormLocationMapStore
present
Flex app gets response, saves formID
check if media items are tagged to questions
no media tagged
Form saved successful msg is shown to the web user
media tagged, post data to FormMediaUploadServlet
Form saved successful msg is shown to the web user
after all media uploaded
Servlet called for each media, saves data and sends resp

Figure 4.3. Activity diagram for form builder module.

Figure 4.4 shows the sequence diagram to explain the flow of control when a new survey form is being created by the administrative user who has access to the system.

Figure 4.5 shows the xml data that is generated by the form builder module for a sample survey form. This generated xml follows the XForms standards.

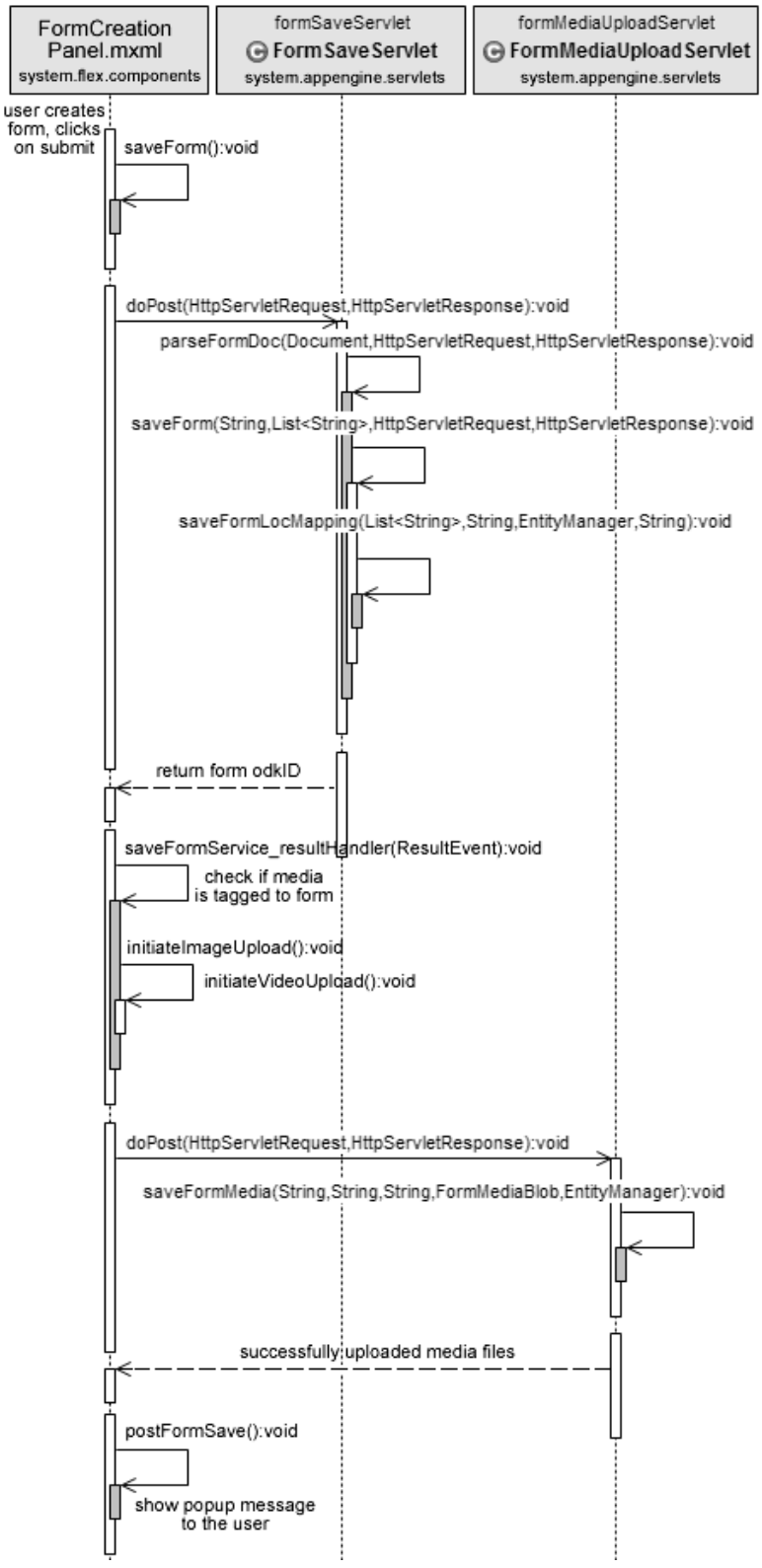


Figure 4.4. Sequence diagram showing the process of saving a form.

```
<?xml version="1.0"?>
<h:html xmlns="http://www.w3.org/2002/xforms"
xmlns:h="http://www.w3.org/1999/xhtml" xmlns:ev="http://www.w3.org/2001/xml-
events" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:jr="http://openrosa.org/javarosa">
  <h:head>
    <h:title>Example Survey Form</h:title>
    <model>
      <instance>
        <model id="exampleSurveyForm" >
          <stringInputType/>
        </model>
      </instance>
      <bind nodeset="/model/stringInputType" required="true()" type="string"/>
    </model>
  </h:head>
  <h:body>
    <input ref="stringInputType">
      <label>Prompt user to enter a string type</label>
    </input>
  </h:body>
</h:html>
```

Figure 4.5. Sample form definition.

4.3.1 Question Attributes

Each of the questions in the form requires the following attributes to successfully define themselves.

1. Label
   This attribute represents the actual text that will be presented as a question to the mobile end user.
2. Hint
   The system has the ability to declare hint to each question which is meant to help the user better understand the question in focus. This field is optional.
3. Is Mandatory
   This field indicates if the question being defined has to be answered mandatorily to complete the survey.
4. Image
   The system supports tagging of an image to each question. If an image is tagged then the user will be presented with that image along with the question. This is optional.
5. Video
   The system also supports tagging of a video to each question. If a video is tagged then the user will be presented with that video along with the question. This is optional.

4.3.2 Question Types

This section introduces all the types of questions that the module currently supports. It also provides a screenshot of how the question type will be displayed to the mobile user and a snippet of how it is defined in the form.

4.3.2.1 STRING TYPE

This question type is the most basic of all the types which accepts only String inputs by the mobile user as a reply. This is very much similar to an input text field we are all familiar with in HTML forms.

The Figure 4.6 shows that only string type input can be entered by the mobile user. When the user touches the text field the inbuilt keyboard of only String type is shown.

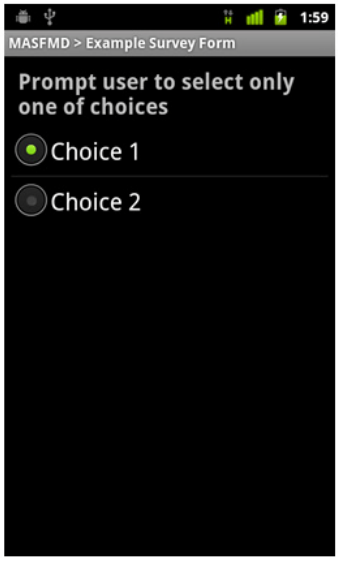
```
<itext>
  <translation lang="English">
    <text id="stringInputType">
      <value form="long">Prompt user to enter a string
type</value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/stringInputType" type="string"/>
.
.
<input ref="stringInputType">
      <label ref="jr:itext('stringInputType')"/>
</input>
```

Figure 4.6 String question type.

4.3.2.2 SINGLE SELECT TYPE

This type of question is used when the user is required to choose one and only one of the already defined answers. It includes a list of options, among which the user can select only one similar to radio buttons in HTML forms.

The Figure 4.7 shows the layout of single select question type. If there are more choices than the screen can show at any time, then the view will be made scrollable which enables the user to scroll through all the available choices.




```
<itext>
  <translation lang="English">
    <text id="singleSelectInputType">
      <value form="long">Prompt user to select only one of
choices</value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/singleSelectInputType" type="select1"/>
.
.
<select1 ref="singleSelectInputType">
      <label ref="jr:itext('singleSelectInputType')"/>
      <item>
        <label>Choice 1</label>
        <value>choice1</value>
      </item>
      <item>
        <label>Choice 2</label>
        <value>choice2</value>
      </item>
</select1>
```

Figure 4.7. Single select question type.

4.3.2.3 MULTI SELECT TYPE

This question type can be used when the user needs to make a choice/choices from the already defined answers. The mobile user is asked to select one or more from the available choices. It contains a list of options which can be selected using checkboxes.

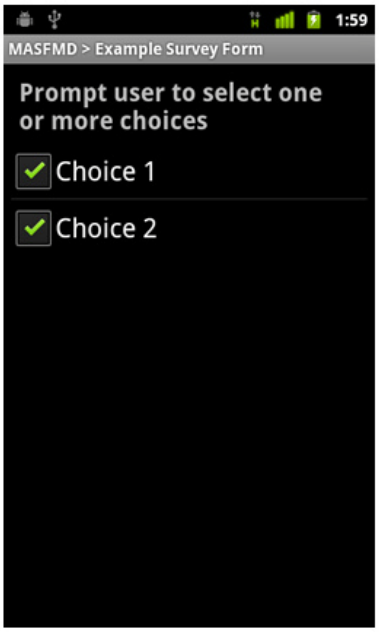
The Figure 4.8 shows the layout of the multi select question type. If there are more choices than the screen can show at any time, then the view will be made scrollable which enables the user to scroll through all the available choices.

4.3.2.4 INTEGER TYPE

The mobile user is asked to type in an integer or number as an answer. Since this is declared as number only field, the user will be allowed to key in only numbers which is accomplished by using the build Android API as shown in Figure 4.9.

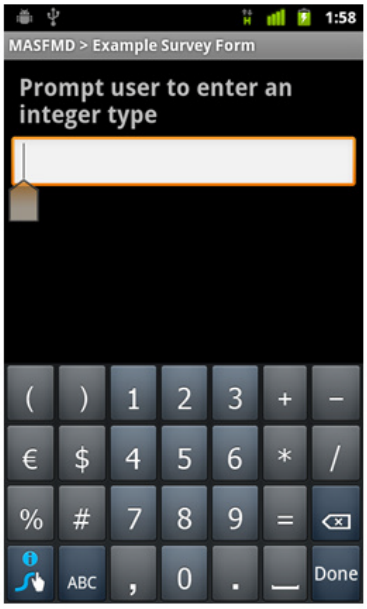
4.3.2.5 GEOLOCATION TYPE

This type of question is used to fetch location information from the mobile user, i.e. geo co-ordinates. To the mobile end user a button is provided which helps in fetching the most accurately available location fix as shown in Figure 4.10.



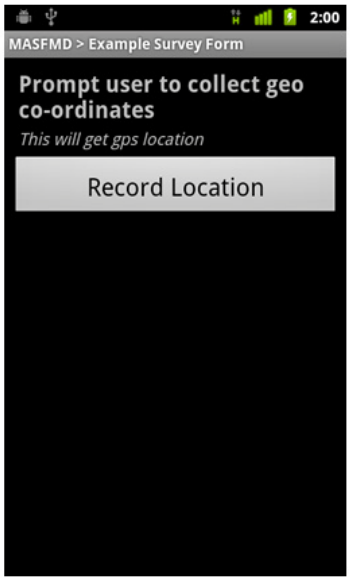

```
<itext>
  <translation lang="English">
    <text id="multiSelectInputType">
      <value form="long">Prompt user to select one or more
choices</value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/multiSelectInputType" type="select"/>
.
.
<select ref="multiSelectInputType">
      <label ref="jr:itext('multiSelectInputType')"/>
      <item>
        <label>Choice 1</label>
        <value>choice1</value>
      </item>
      <item>
        <label>Choice 2</label>
        <value>choice2</value>
      </item>
</select>
```

Figure 4.8. Multi select type question.




```
<itext>
  <translation lang="English">
    <text id="integerInputType">
      <value form="long">Prompt user to enter an integer
type</value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/integerInputType" type="int"/>
.
.
<input ref="integerInputType">
      <label ref="jr:itext('integerInputType')"/>
</input>
```

Figure 4.9. Integer type question.




```
<itext>
  <translation lang="English">
    <text id="geopointInputType">
      <value form="long">Prompt user to collect geo co-
ordinates</value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/geopointInputType" type="geopoint"/>
.
.
<input ref="geopointInputType">
      <label ref="jr:itext('geopointInputType')"/>
      <hint>This will get gps location</hint>
</input>
```

Figure 4.10. Geolocation type question.

4.3.2.6 IMAGE TYPE

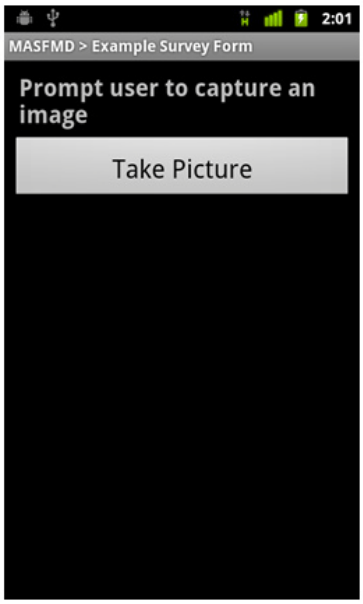
This type prompts the user to capture an image from the device's built-in camera. There is ability to retake, delete images taken in the scenario where-in the user is not satisfied with the picture taken. Figure 4.11 shows the user interface for this of question on the mobile application.

4.3.2.7 AUDIO TYPE

This type of question asks the user to record an audio clip using the default microphone recorder application on the device. As with the image capture, the user has options to retake, delete and review the audio recording. Figure 4.12 shows the user interface for Audio type question on the mobile application.

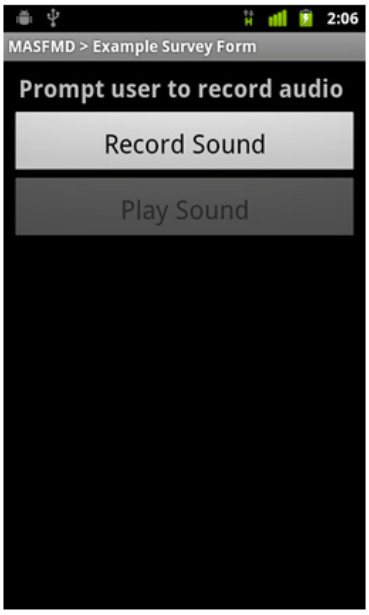
4.3.2.8 VIDEO TYPE

This type of question prompts the user to capture a video using the built-in camera. There is ability to retake, delete video already taken in the scenario where-in the user is not satisfied or changes his mind. Figure 4.13 shows the user interface for Video type question on the mobile application.



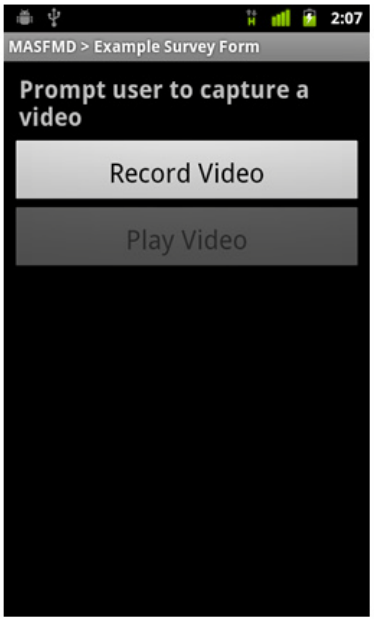

```
<itext>
  <translation lang="English">
    <text id="imageInputType">
      <value form="long">Prompt user to capture an
image</value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/imageInputType" type="binary"/>
.
.
<upload ref="imageInputType" mediatype="image/*">
      <label ref="jr:itext('imageInputType')"/>
</upload>
```

Figure 4.11. Image type question.




```
<itext>
  <translation lang="English">
    <text id=" audioInputType ">
      <value form="long"> Prompt user to record audio </value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/audioInputType" type="binary"/>
.
.
<upload ref="audioInputType" mediatype="audio/*">
      <label ref="jr:itext('audioInputType')"/>
</upload>
```

Figure 4.12. Audio type question.




```
<itext>
  <translation lang="English">
    <text id="videoInputType">
      <value form="long">Prompt user to capture a video</value>
    </text>
  </translation>
</itext>
.
.
<bind nodeset="/model/videoInputType" type="binary"/>
.
.
<upload ref="videoInputType" mediatype="video/*">
      <label ref="jr:itext('videoInputType')"/>
</upload>
```

Figure 4.13. Video type question.

4.3.2.9 IMAGE TAGGED TO QUESTIONS

The form builder also supports images to be tagged to each of the above type of questions. Each of these images is stored as blob data type on the Google App Engine datastore.

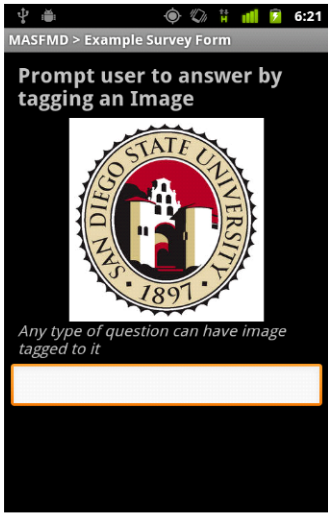
The Figure 4.14 shows the user interface on the mobile application. If the user clicks or touches the image the image is opened using the default image viewer specified for that device, which could help the user to inspect the image in more detail.

4.3.2.10 VIDEO TAGGED TO QUESTIONS

The form builder also supports videos to be tagged to each of the above type of questions. These tagged videos are stored as blob data type on the Google App Engine datastore.

The Figure 4.15 shows the user interface on the mobile application. If the user clicks or touches the play icon it is opened using the default video player specified for that device.

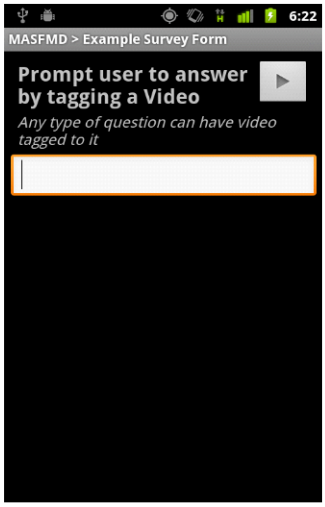
We have just seen how multimedia can be tagged to each of the question types. Also it is to be noted that both multimedia types, i.e. image and video can be tagged to each of the above question types making the question more fun to answer by giving visual clues to the user. After all the questions have been created by the administrator of the system or a user


6:21
MASFMD > Example Survey Form
Prompt user to answer by tagging an Image
SAN DIEGO STATE UNIVERSITY
1897
Any type of question can have image tagged to it

```
<itext>
  <translation lang="English">
    <text id="imageTagType">
      <value form="long">Prompt user to answer by tagging an
image</value>
      <value form="image">jr://images/question1.jpeg</value>
      <value form="big-image">jr://images/question1.jpeg</value>
    </text>
  </translation>
</itext>
.
.
.

<input ref="imageTagType">
      <label ref="jr:itext('imageTagType')">
      <hint>Any Type of question can have image tagged to it</hint>
</input>
```

Figure 4.14. Image tagged to question.




```
<itext>
  <translation lang="English">
    <text id="videoTagType">
      <value form="long"> Prompt user to answer by tagging a
video</value>
      <value form="video">jr://video/question1.3gp</value>
    </text>
  </translation>
</itext>
.
.
.

<input ref="videoTagType">
      <label ref="jr:itext('videoTagType')">
      <hint>Any Type of question can have video tagged to it</hint>
</input>
```

Figure 4.15. Video tagged to question.

who has access to the system, he then has to associate or tag it to a location. When the user clicks on the “Tag Locations” button, he is shown a list of all the locations that are currently available on the system so he can choose the appropriate one.

4.4 Location Builder Module

This module provides an interface for the user to create and/or edit locations on the system. It supports all the available features of Google maps, namely:

- Options to change the map type, i.e. Map, Satellite, Hybrid, Terrain.
- Zoom in, Zoom out
- Map panning

Each location in the system has the following attributes:

1. Name:
   This is the user viewable name of the location
2. Geo Co-ordinates:
   This includes the latitude and the longitude of the location.
3. Radius:
   Each location has an alert radius. This radius is used to determine if the mobile user is within its purview and hence if the user should be alerted to take part in the survey.
4. FormID:
   A location can have zero or at most one form tagged to it.

Figure 4.16 shows the mapping of location objects with that of the form. It also shows the schema of the form datastore, location datastore and how their respective media are tied to them.

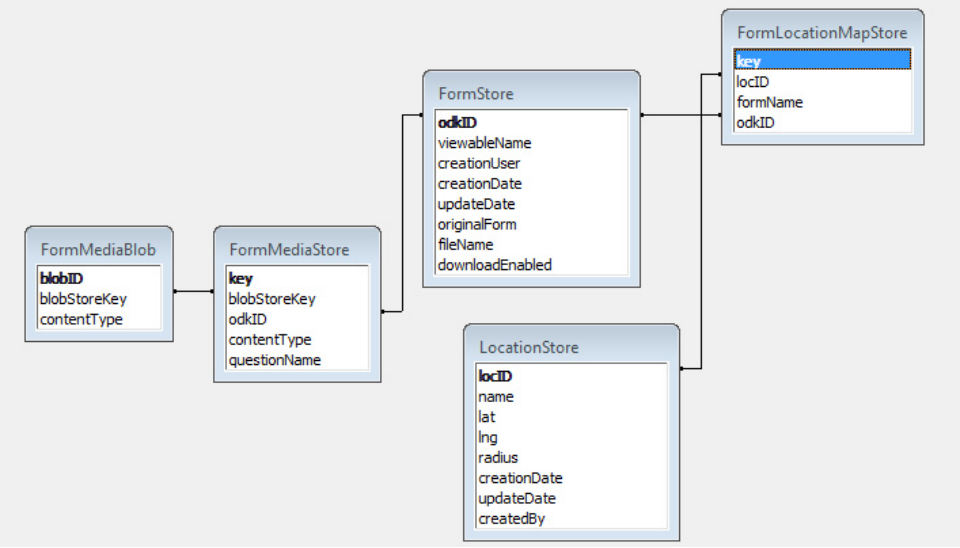


Figure 4.16. Database mapping for location and form module.

4.5 FORM SUBMISSION VIEWER MODULE

Once the form is created and the mobile users start submitting the survey answers, the need for visual representation of this data arises to better help analysis of available data. This

module provides a user interface to analyze the slew of submissions in an organized, graphical manner.

The Figure 4.17 shows the activity flow when the user requests to view the submission data on the flex web application. All the data that is stored on the server is retrieved by requesting it thought servlets and waiting for the response from them. When the *FormSubmissionsXMLServlet* gets to process the request and *odkID* of the form is passed as a parameter in the request. Using this *odkID* the result table which contains all the submission data is formed and retrieved from the datastore. This is later converted into an xml string which is friendly in nature with the build flex application when then uses it to render the submission viewer. Depending on the type of the question, different representations are rendered.

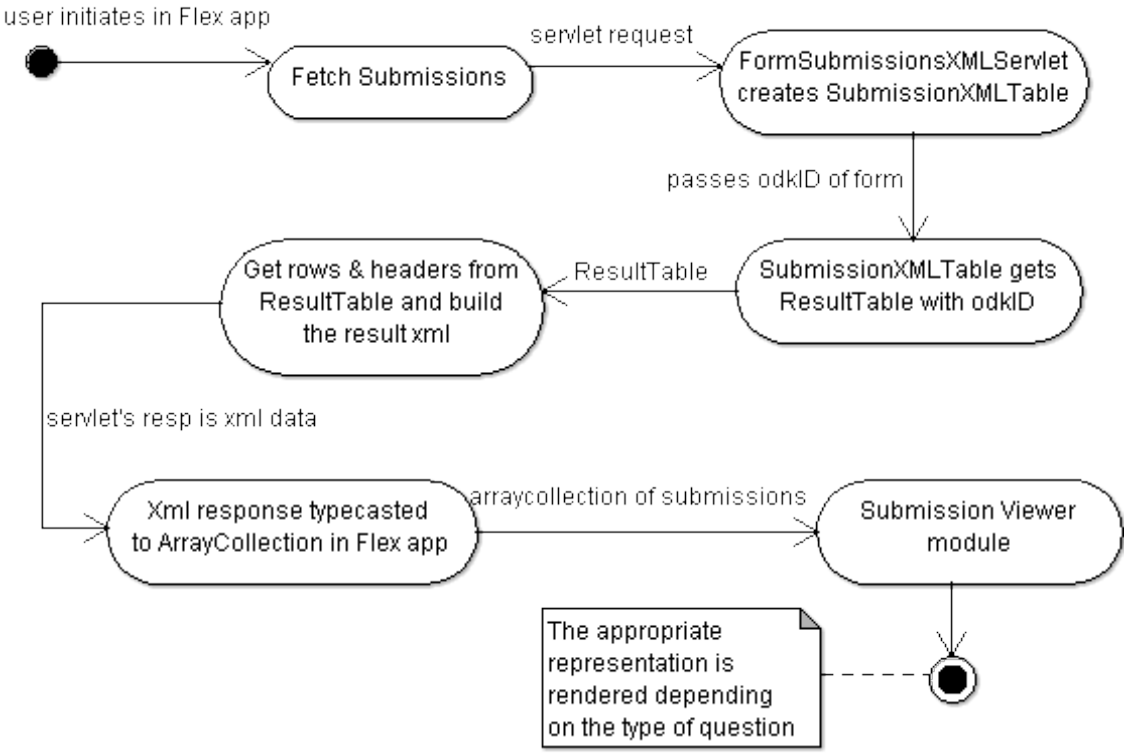

user initiates in Flex app
Fetch Submissions
servlet request
FormSubmissionsXMLServlet creates SubmissionXMLTable
passes odkID of form
SubmissionXMLTable gets ResultTable with odkID
ResultTable
Get rows & headers from ResultTable and build the result xml
servlet's resp is xml data
Xml response typecasted to ArrayCollection in Flex app
arraycollection of submissions
Submission Viewer module
The appropriate representation is rendered depending on the type of question

Figure 4.17. Activity diagram for submission viewer module.

The Figure 4.18 shows the sequence diagram for the submission viewer process.

The Figure 4.19 shows the class diagram for *SubmissionXMLTable* which is responsible for fetching the *resultTable* of the form. The following section describes in detail the different types of representations for viewing the submission data.

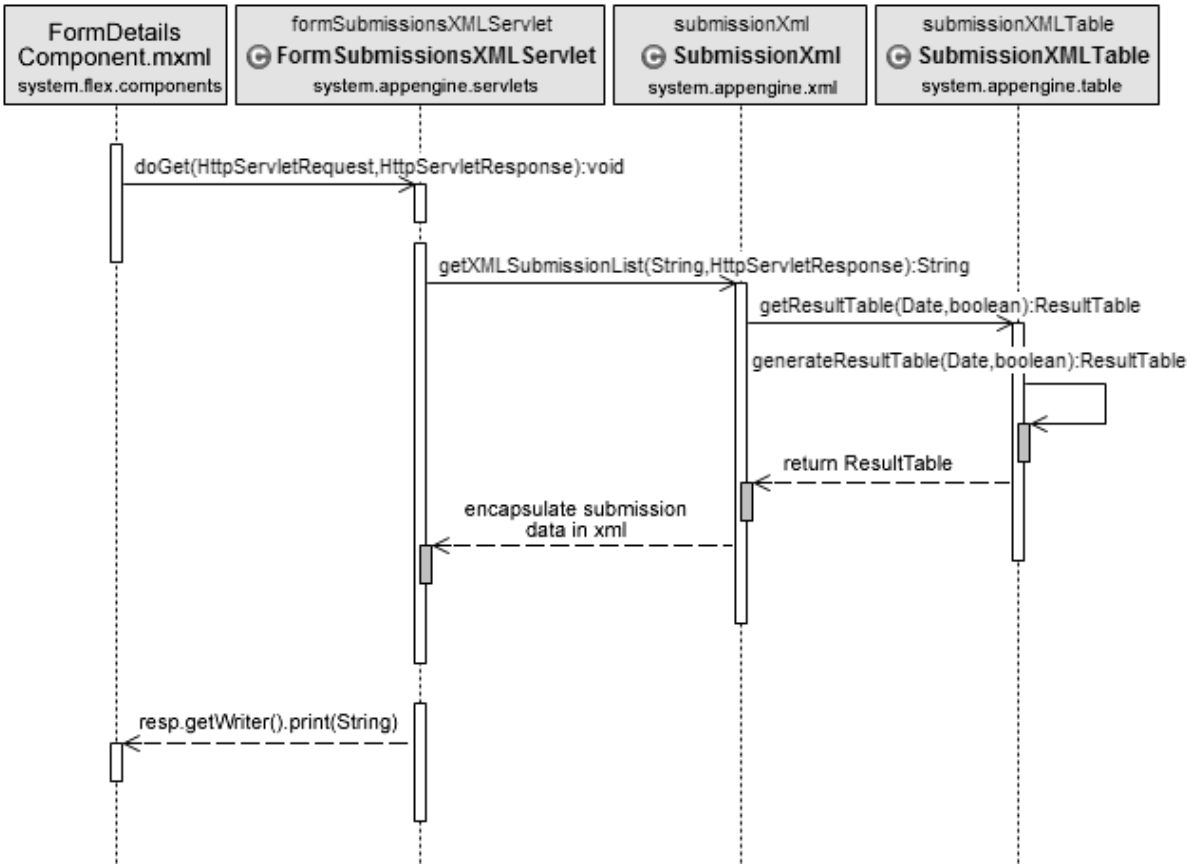


Figure 4.18. Sequence diagram for submission viewer process.

4.5.1 String Representation

This is the simplest form of representation available on the system. It consists of a list of strings ordered chronologically on submission date as shown in Figure 4.20.

Question types using this representation:

1. String type

This form of representation consists of 3 types of graphs/charts:

1. Pie Chart
2. Bar Chart
3. Column Chart

4.5.2 Graphical Representation

Question types using this representation:

1. Multi Select type

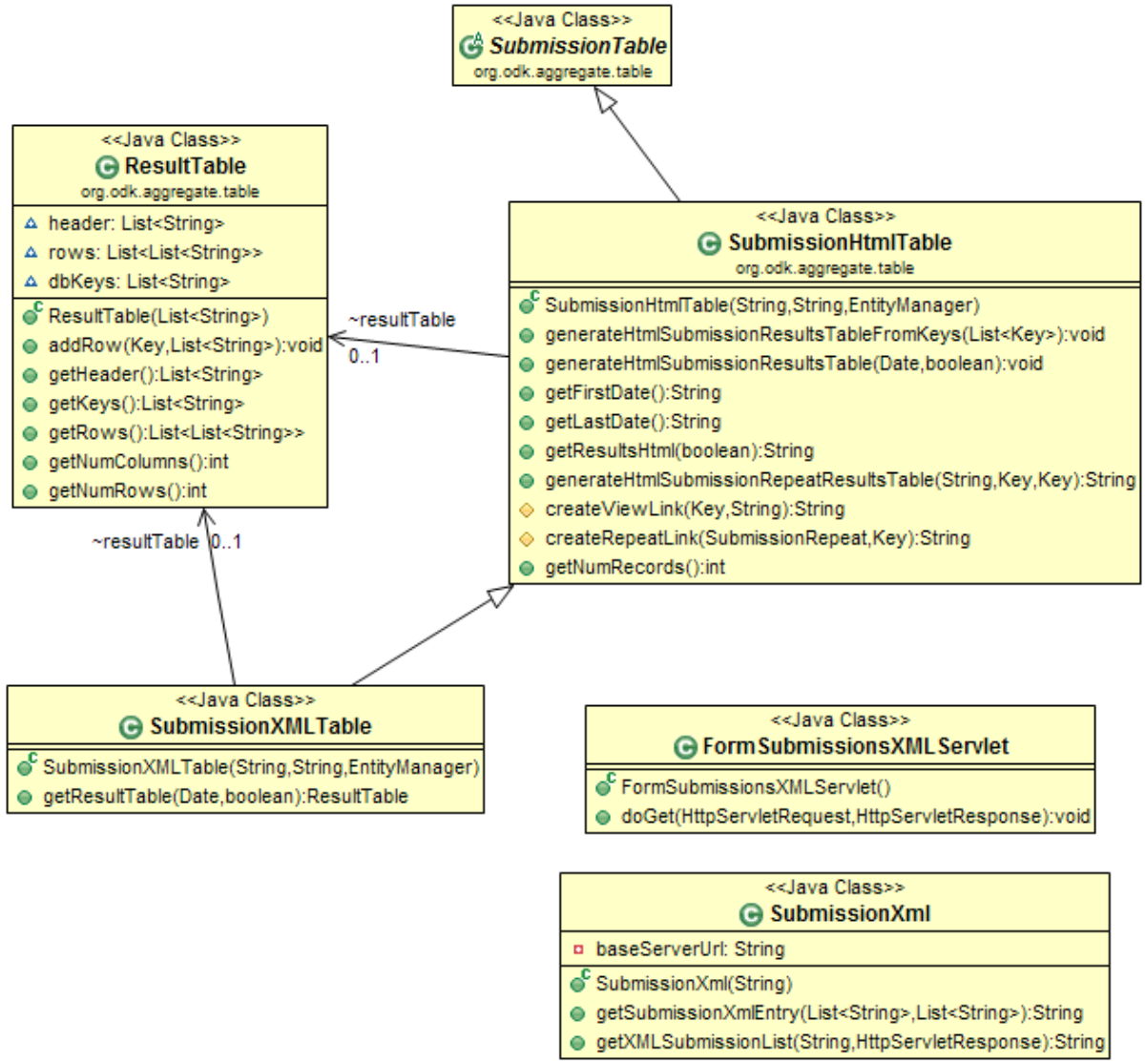


Figure 4.19. Class diagram for SubmissionXMLTable.

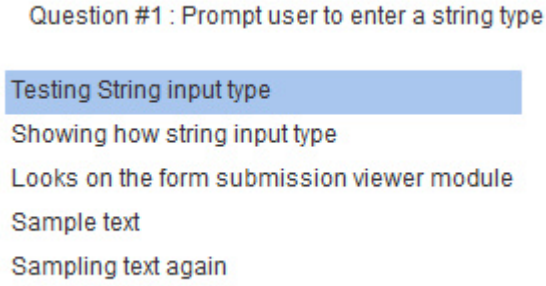

Question #1 : Prompt user to enter a string type
Testing String input type
Showing how string input type
Looks on the form submission viewer module
Sample text
Sampling text again

Figure 4.20. String representation for submission.

2. Single Select type
3. Integer type

Figure 4.21 shows how a summary of data is presented to the viewer who has an option to dynamically change the type of graph he wants to view by clicking on the desired "Chart" button.

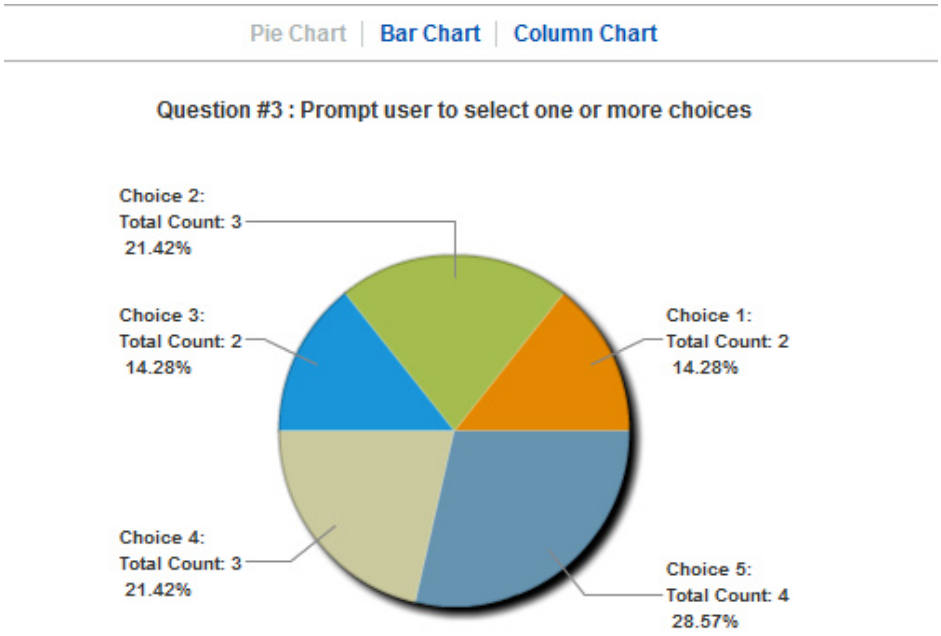


Figure 4.21. Graphical representation for submission.

4.5.3 Map Representation

This form of representation uses Google Maps to show all the submission entries. Each entry is drawn as a marker on the map using the geo co-ordinates, i.e. latitude and longitude. It supports all the available features of Google maps, namely

- Options to change the map type, i.e. Map, Satellite, Hybrid, Terrain.
- Zoom in, Zoom out
- Map panning

Geolocation type of question uses this representation as shown in Figure 4.22.

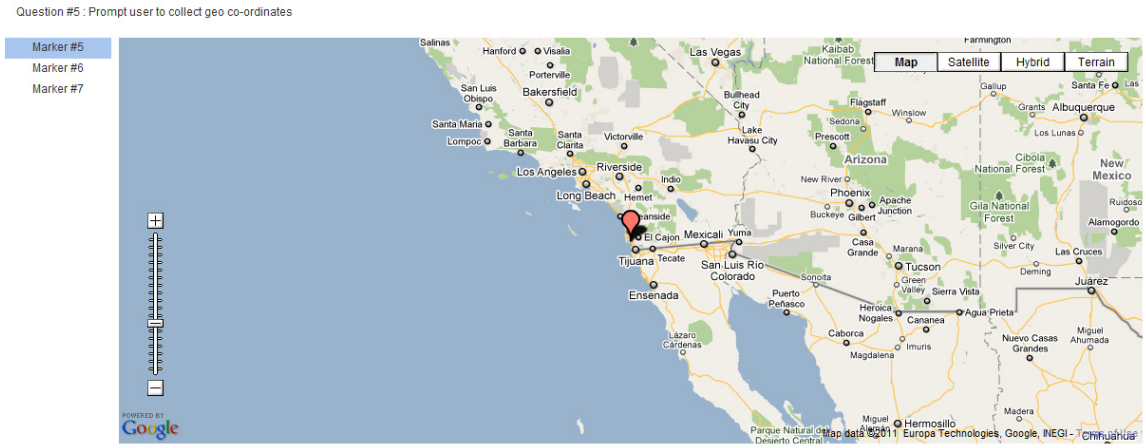


Figure 4.22. Map representation for submission.

4.5.4 Image Viewer Representation

This provides a visual representation of the images uploaded by the mobile users as part of the survey. All the images are rendered on a carousel interface, ordered chronologically on submission date as shown in Figure 4.23

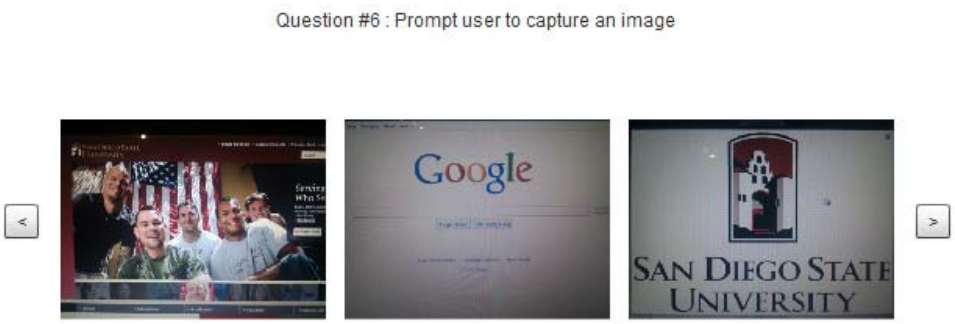


Figure 4.23. Image viewer representation for submission.

This type of representation is used only by the Image Type of questions.

4.5.5 Media Player Representation

This provides an interface to evaluate media file types uploaded by the mobile users as part of the survey. All the associated media items are listed and upon clicking each of them the appropriate media player is displayed to view and/or hear the media.

Question types using this representation:

1. Audio type
2. Video type

Figure 4.24 shows how media type submission is shown to the user. It consists of a list of available media on the left hand column and the player on the right hand side. Each of these media files can be viewed by clicking on the desired item in the list, which triggers the player to stream the chosen media.

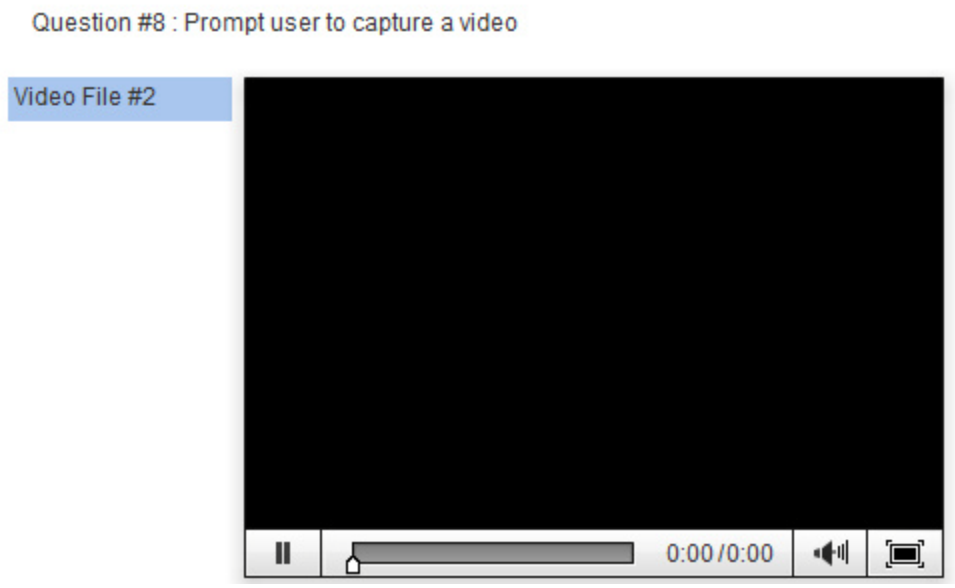


Figure 4.24. Media player representation for submission.

4.6 DATA CHANGE NOTIFICATION MODULE

One of the key challenges with the project was how to notify the mobile user's device when the data on the server has changed without using the age old polling technique, which as explained earlier is not efficient. Hence C2DM framework is used to communicate between the server and connected devices. GAE's Task Queue API is used to send these messages to all the connected devices.

4.6.1 What is a Task Queue API

This API allows applications to perform work outside of a user request, initiated by a user request. Whenever an app needs to execute background tasks, GAE allows us to use this API to organize work into small, discrete units called Tasks. The app adds tasks to task queues to be executed later.

In the event of any update, delete or creation of a location, C2DM enveloped message is broadcast to all the devices registered on the server. To broadcast this message the server

creates a task to fetch all the connected devices and then sends the message to each one.

The Figure 4.25 shows the activity diagram of how the data change notification module works after the user has made some changes to the system using the web application.

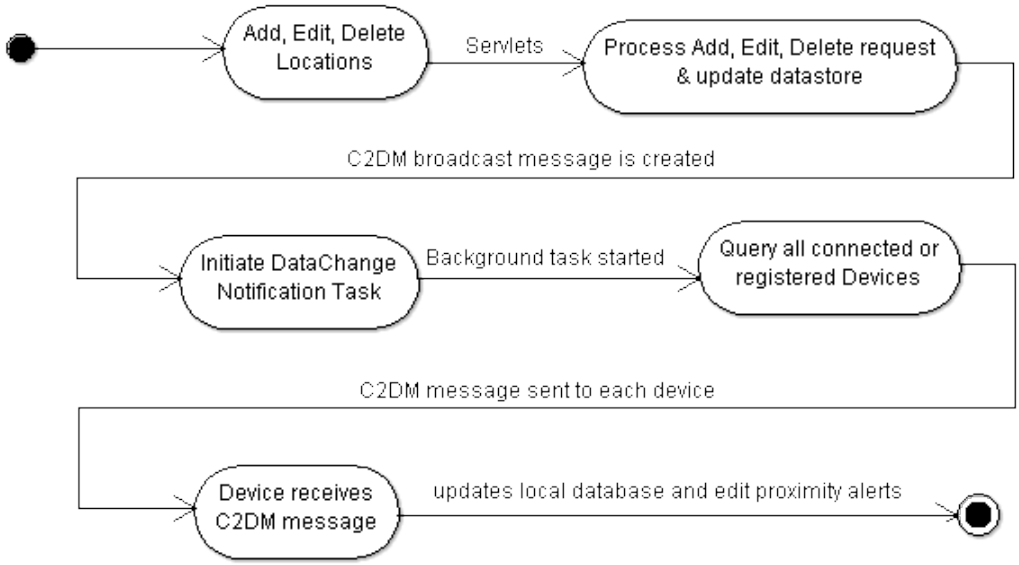

Add, Edit, Delete Locations
Servlets
Process Add, Edit, Delete request & update datastore
C2DM broadcast message is created
Initiate DataChange Notification Task
Background task started
Query all connected or registered Devices
C2DM message sent to each device
Device receives C2DM message
updates local database and edit proximity alerts

Figure 4.25. Data change notification activity diagram.

The Figure 4.26 shows the sequence diagram for the data change notification module. It shows how the task is created and how the control flows in the system upon change of data.

Queue queue = QueueFactory.getQueue("send-to-all-devices");
queue.add(TaskOptions.Builder.url("/sendToAll").param("msg",msg));

The above snippet of code shows how to instantiate broadcast task and add it to the Task Queue where:

"msg" - is the C2DM that needs to be broadcast. To see the structure of the C2DM message please refer below:

"send-to-all-devices" refers to a particular queue entry in the queue.xml that needs to be created in the /war/WEB-INF folder. The structure of the xml file is shown in Figure 4.27.

4.6.2. Structure of C2DM Message

The C2DM messages are sent from the server to all the connected devices indicating the nature of the data change on the server. Depending on the attributes set in this C2DM message, the client devices make appropriate changes on the local database.

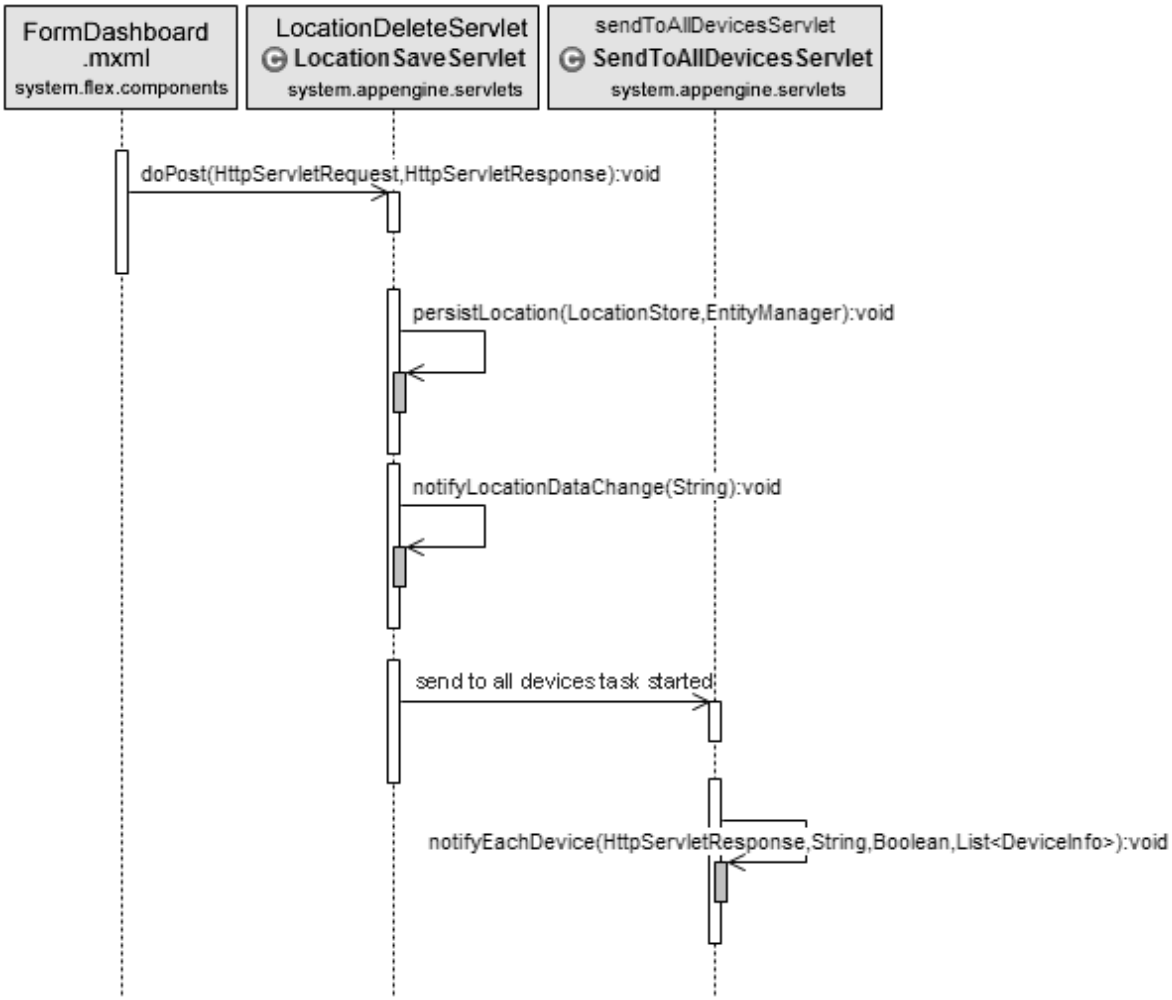


Figure 4.26. Sequence diagram for data change notification.

```
<?xml version="1.0" encoding="utf-8"?>
<queue-entries>
  <queue>
    <name>c2dm</name>
    <rate>10/s</rate>
  </queue>
  <queue>
    <name>send-to-all-devices</name>
    <rate>10/s</rate>
  </queue>
</queue-entries>
```

Figure 4.27. Structure of queue.xml.

The Figure 4.28 shows the structure of a sample message which is broadcast to all the devices using C2DM framework. Each message has the following attributes:

- action:
  This specifies the type of action that needs to be performed by the client device.
- actionID:
  Contains the ID of the action that needs to be carried out.
- actionName:
  Human readable name of the item on which the action is being performed.
- requestTimestamp:
  It contains the timestamp created by the server at which this action was requested.

```
<c2dm
action="formFetch"
actionID="exampleSurveyForm"
actionName="Example Survey Form"
requestTimestamp="1307261108630"/>
```

Figure 4.28. Structure of sample C2DM message.

The different types of actions that the system uses to communicate with the connected mobile devices are:

- *formFetch (Notifiable message)*
  Instructs the device that the user is requested to be part of a survey having the formID specified in the message attribute “actionID”.
- *locationUpdate (Non-Notifiable message)*
  Instructs the device to update the existing location having the locationID specified in the message attribute “actionID”. Also the attribute “actionName” indicates the human-readable name of the location.
- *locationAdd (Non-Notifiable message)*
  Instructs the device to add a new location having the locationID specified in the message attribute “actionID”.
- *locationDelete (Non-Notifiable message)*
  Instructs the device to delete the location having the locationID specified in the message attribute “actionID”.

For more information about Notifiable and Non-Notifiable message please refer section Location Service Module.

4.7 SUBMISSION DATA EXPORT MODULE

The main purpose of this application is to help the user to analyze data using a rich, intuitive driven user interface. This system also supports the feature to enable users to export the gathered submission data from the mobile users to an excel sheet.

The Figure 4.29 shows the activity diagram for the survey exporting feature. This feature is already a part of ODK's aggregate tool and it has been used to save the submission data to a .csv file. This facilitates the user to analyze the data in any other manner he desires, without having to be restricted to the ones represented by this system. For more information please refer section "How to export submission" in the User Manual.

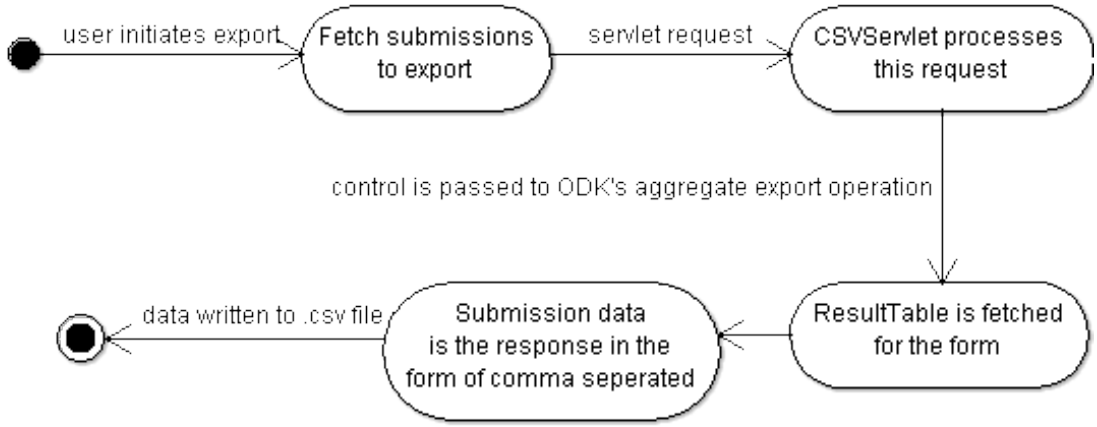

user initiates export
Fetch submissions to export
servlet request
CSVServlet processes this request
control is passed to ODK's aggregate export operation
ResultTable is fetched for the form
Submission data is the response in the form of comma seperated
data written to .csv file

Figure 4.29. Activity diagram for survey export module.

The Figure 4.30 shows the sequence diagram for the process of exporting submission data from the system, which is then written onto a csv file.

4.8 ADMIN MODULE

When any user navigates to the application by typing in the website address does not have access to the system, unless he is the administrator. Hence that general user only sees the welcome page after he logs in using a Gmail account. This module provides the administrator of the momentary assessment system functionality to provide desired specific users access to the application.

In order for the administrator to provide access to the desired user, he has to add that user's emailID into the system using this module and grant him permission to use the system.

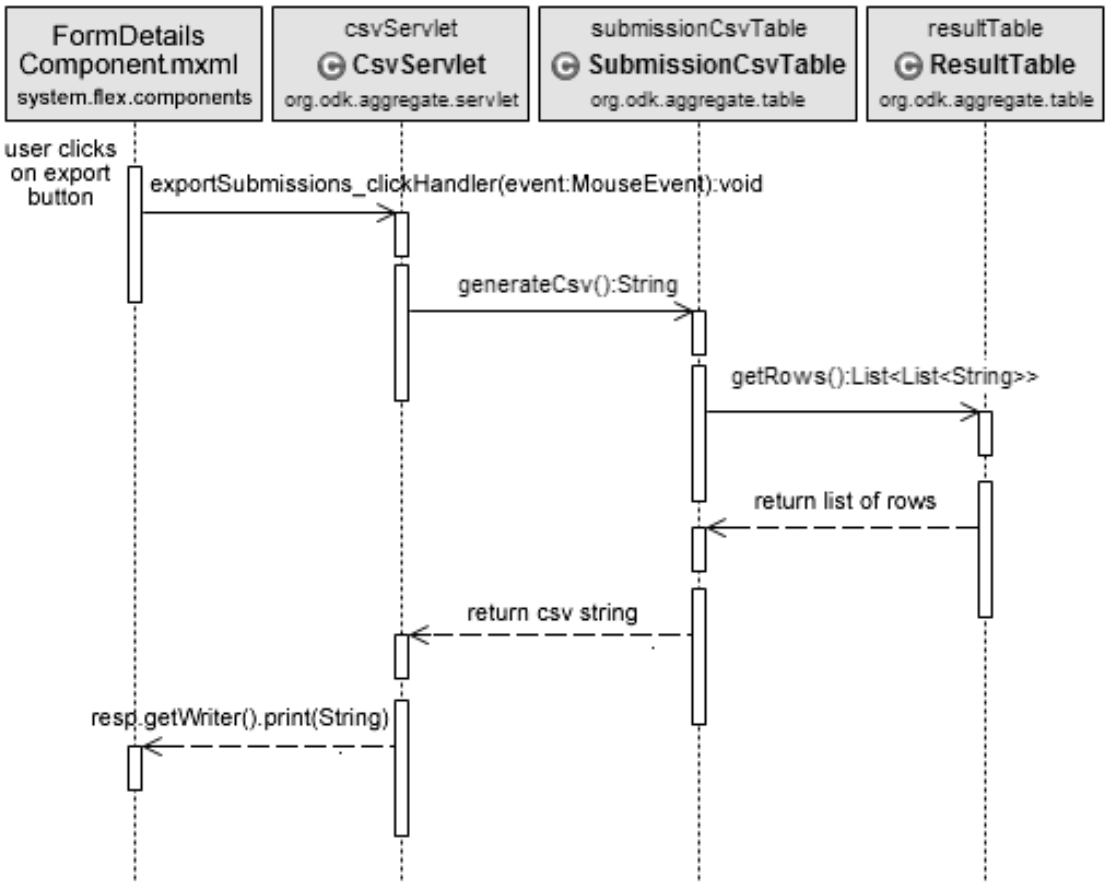


Figure 4.30. Sequence diagram for exporting submission data.

The access to the system is regulated using the Users service provided by the Google App Engine java, which lets the application integrate with Google user accounts. With the Users service, users can use the Google accounts they already have to sign in to the application. The Figure 4.31 shows a sample code on how to use this service.

```
UserService userService = UserServiceFactory.getUserService();
User user = userService.getCurrentUser();

if (user != null) {
   // user has logged in
} else {
  // user has not logged in, hence send login url.
  resp.sendRedirect(userService.createLoginURL(req.getRequestURI()));
}
```

Figure 4.31. Sample Google app engine user service code.

4.9 LOCATION SERVICE MODULE

This module is responsible for listening to any data change notification that is sent by the server when there is any change in the system, as explained in section Data Change notification module. When the mobile client receives these messages, it wakes up the application's background location service module and indicates that something on the server side has changed. This is responsible for registering all the updates that the device receives from the server and takes appropriate action based on the nature of the message sent.

The Figure 4.32 shows the class diagram for *LocationService* which is the background service responsible for keeping the location data updated on the mobile client device. The operations carried out by this service are determined by the nature of intent with which it was started. The operations are defined primarily by the C2DM messages sent from the server to all the connected mobile devices.

As show in Figure 4.33 C2DM Messages are broadly classified as:

- Notifiable messages
  The messages which require UI or foreground notifications come under this category. An example of this type can be when the server broadcasts a message to a device explicitly asking the user to be a part of a survey upon which the application shows an UI notification using the Android notification manager.
- Service (Non-Notifiable) messages
  These messages does not require any user intervention to complete the requested operation successfully. An example can be when there is a change in the attributes of a particular location the server broadcasts a message to all the devices and asks them to update the changed attribute.

The Location Service module also ensures that the mobile client device is in sync with the data present on the server at any given time. The Figure 4.34 shows the sequence diagram to explain the process.

4.10 LOCATION ALERT MODULE

This module is responsible for detecting if the user (mobile device) is physically present within the desired range of location. As defined earlier each location has the geo co-ordinates and the alert radius as its attribute. A local copy of this information is stored on the mobile device's database.

When the user is connected to the service, it contacts the server to fetch all the locations and sets proximity alerts to those existing locations. These alerts are responsible for

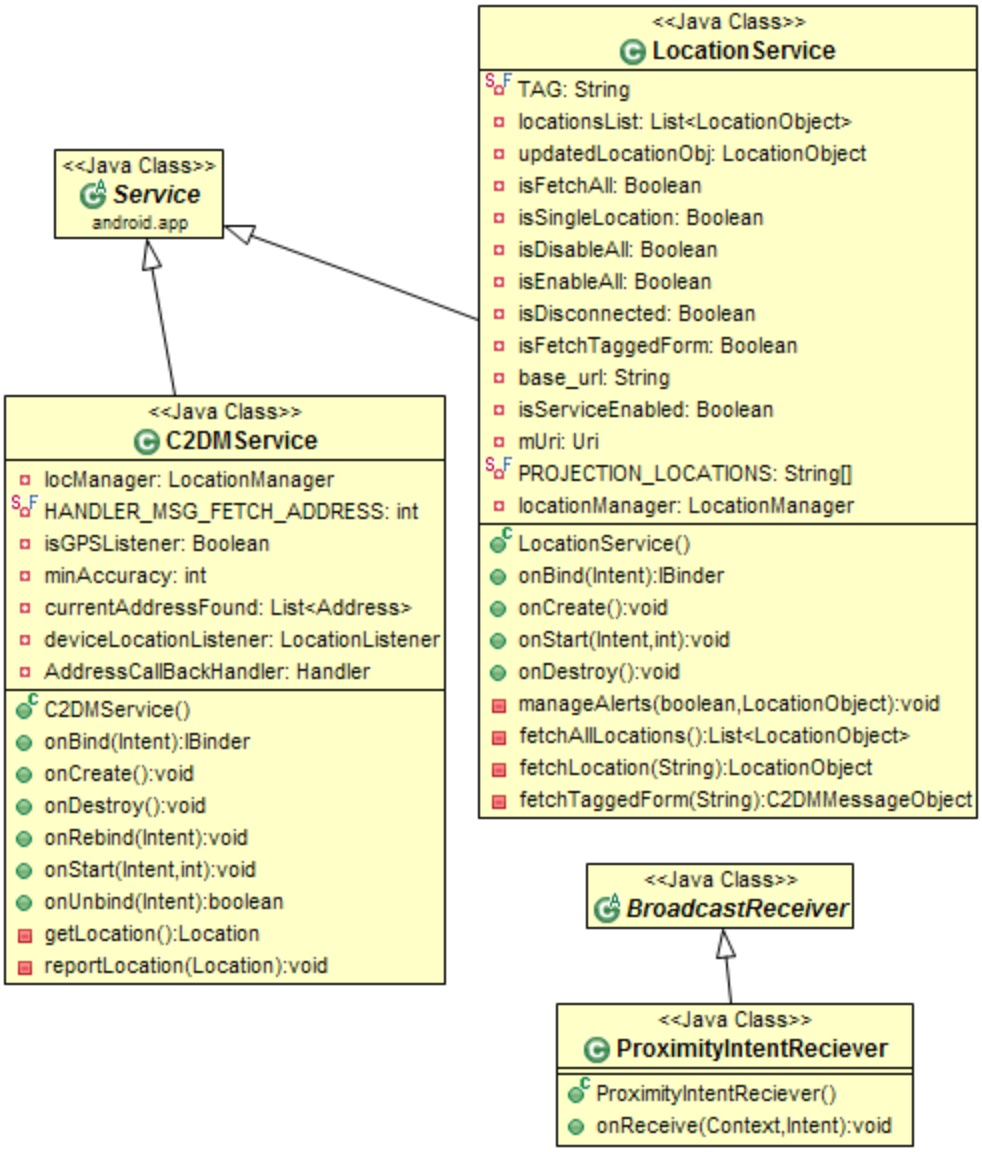


Figure 4.32. Class diagram for LocationService.

notifying the mobile application that the user has entered into a region of importance and that certain action needs to be performed. This module works in conjunction with the Location Service module to maintain the system on the client device, making sure that all updates and edits to already existing locations are made to reflect on the device.

The Figure 4.35 explains how the module works. Initially when the user has just registered to the system, all the available locations are fetched from the server and a local copy is saved. Then a proximity alert is set to each of those locations.

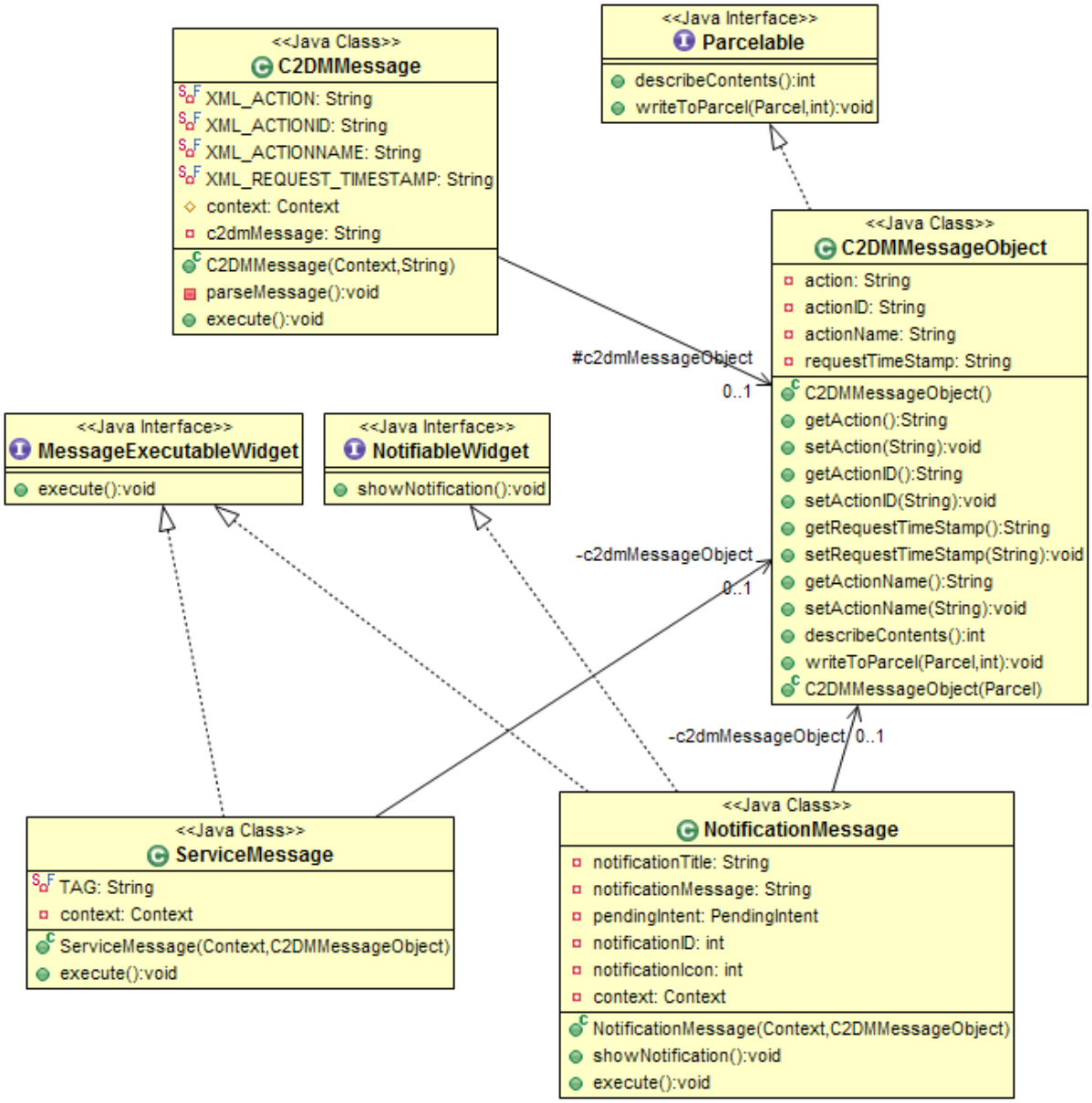


Figure 4.33. Class diagram for C2DM message.

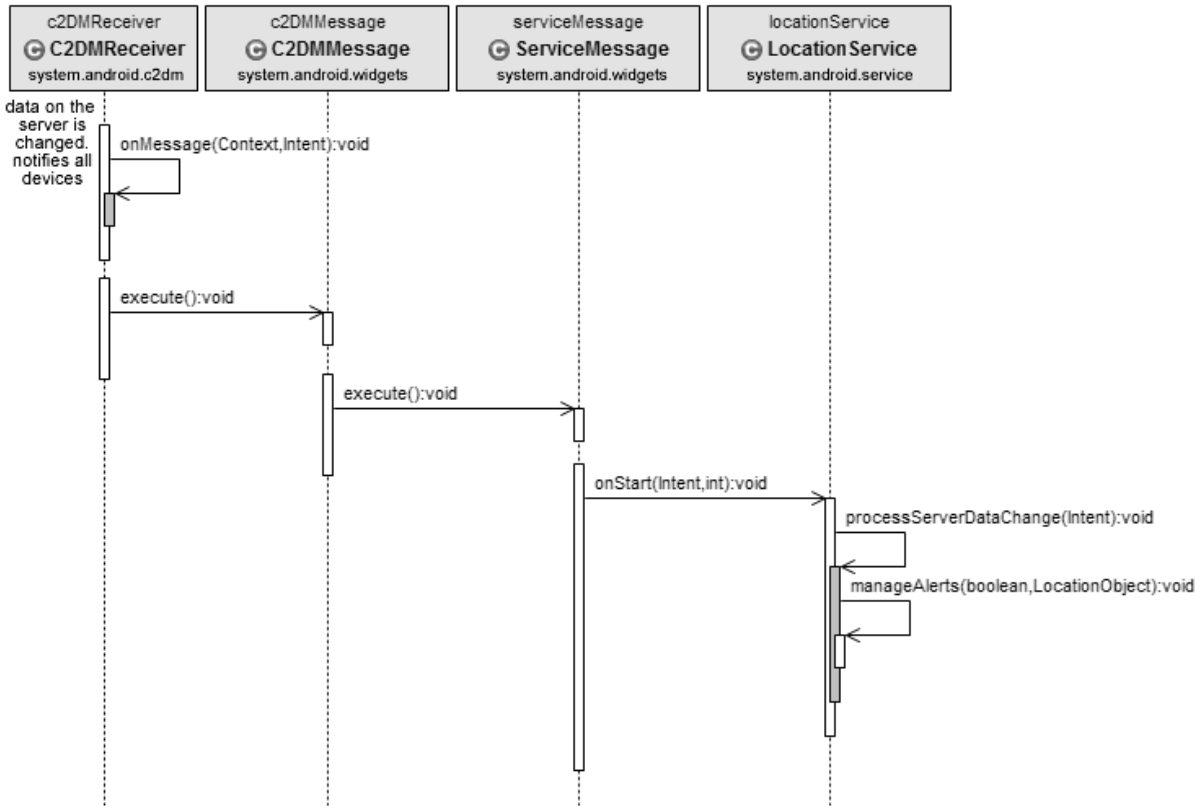


Figure 4.34. Sequence diagram explaining the location service module.

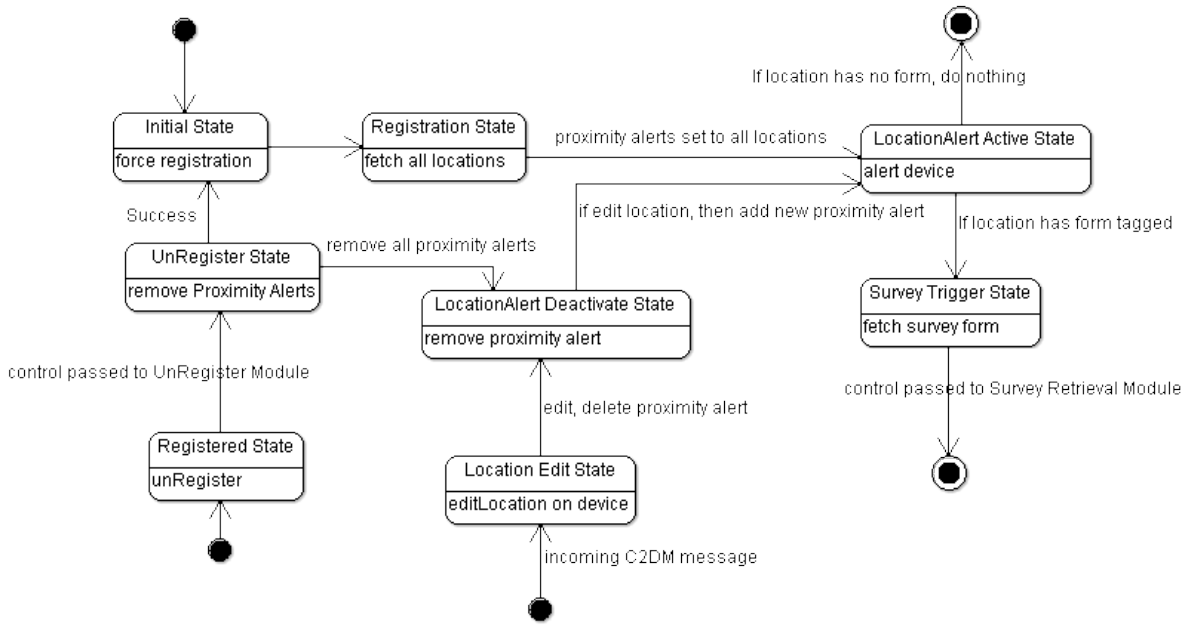


Figure 4.35. State diagram for location alert module.

The Figure 4.36 shows the sequence diagram explaining how the location alerts are set from the time the user registers to the system and the role of *ProximityIntentReceiver*.

Setting location alerts on the mobile device: Proximity alerts are set for all the locations that are found in the database using the Google Android built-in proximity alert API provided by the *LocationManager* class.

addProximityAlert (double latitude, double longitude, float radius, long expiration, PendingIntent intent)

This method sets a proximity alert for the location given by the position (latitude, longitude) and the given radius. When the device detects that it has entered or exited the area surrounding the location, the given PendingIntent will be used to create an Intent to be fired. The fired Intent will have a boolean extra and if the value is true, the device is entering the proximity region; if false, it is exiting.

Due to the approximate nature of position estimation, if the device passes through the given area briefly, it is possible that no Intent will be fired. Similarly, an Intent could be fired if the device passes very close to the given area but does not actually enter it.

Parameters required by this API:

- latitude
  the latitude of the central point of the alert region
- longitude
  the longitude of the central point of the alert region
- radius
  the radius of the central point of the alert region, in meters
- expiration
  time for this proximity alert, in milliseconds, or -1 to indicate no expiration
- intent
  a PendingIntent that will be used to generate an Intent to fire when entry to or exit from the alert region is detected

Removing location alerts on the mobile device: Similar to adding proximity alerts, Google Android built-in API is used to remove the proximity alerts.

removeProximityAlert (PendingIntent intent)

This method removes the proximity alert with the given PendingIntent. This PendingIntent has to be the exact same as used when adding the proximity alert.

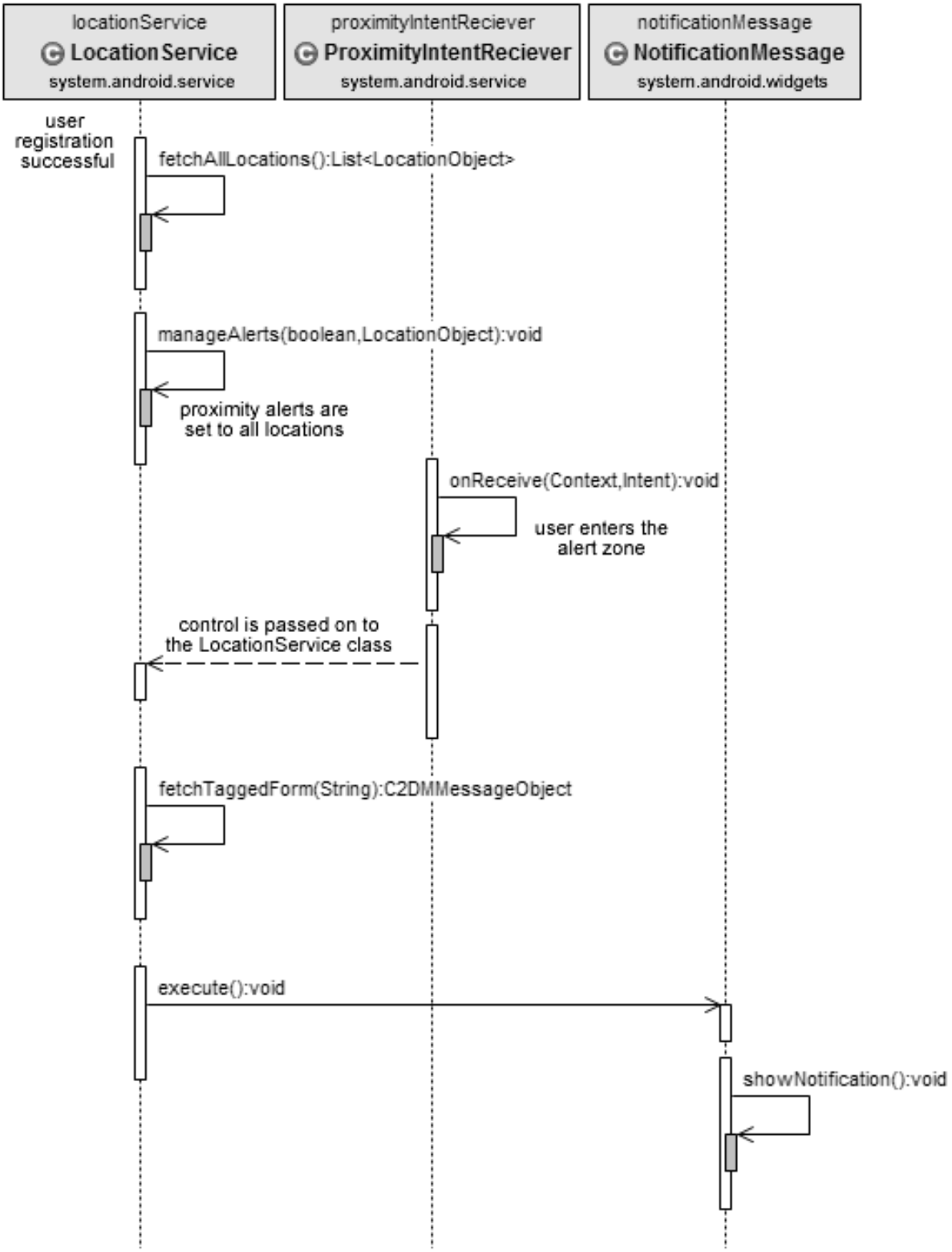


Figure 4.36. Sequence diagram explaining the location alert process.

Hence *locationID* is currently used as the unique qualifier to identify these pending intents. This PendingIntent will perform a broadcast and receivers which are registered to listen to this broadcast are awakened.

ProximityIntentReciever is a class which implements *BroadcastReceiver* which listens to the broadcast messages. It checks to see if the user has entered or exited the proximity region. If the user has entered then the control is passed to the *LocationService* class to check if a form is tagged to this location. The service then contacts the server to get the form details.

If a form is tagged then the *NotificationMessage* class is executed to show a UI notification to the user, indicating that he has been requested to take part in a survey. In case there is no form tagged to that particular location then the alert is dismissed.

In the event of a location update C2DM message that is sent from the server to the connected devices, then the Location Service module first removes the existing proximity alert for that particular location. Then edit the information for that location on the device, and then adds a new alert with the changed location values as shown in Figure 4.29

4.11 SURVEY RETRIEVAL MODULE

This module is responsible for downloading the form and allowing the user to take the survey. The Figure 4.37 shows the activity diagram for this whole process. Once the user has been notified to take part in a survey and the user clicks on the notification then the form content has to be fetched from the server. A new task is spawned to fetch the form. Android AsyncTask is used to implement these tasks.

The Figure 4.38 shows the sequence diagram for explaining how the forms are downloaded, once the user accepts to be part of the survey by clicking on the Continue button. After the completion of fetching the form, the control is passed on to ODK's *FormEntryActivity*.

The Figure 4.39 shows the class diagrams of the tasks involved in the survey form retrieval process.

DownloadFormTask: This is an *AsyncTask* responsible for fetching the form using the *formID* provided by the location alert module and saves the local copy of the form.xml. Then the module contacts the server to check if there are any media files (i.e. image, audio, video

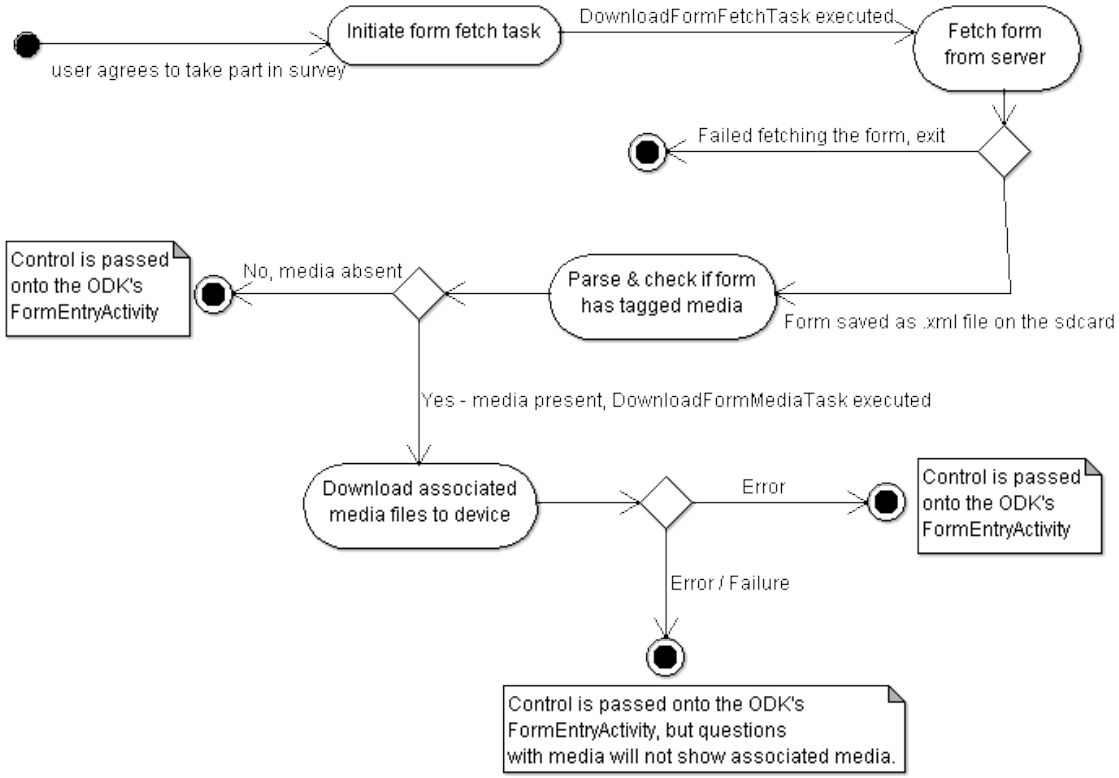

Initiate form fetch task
DownloadFormFetchTask executed
Fetch form from server
user agrees to take part in survey
Failed fetching the form, exit
Control is passed onto the ODK's FormEntryActivity
No, media absent
Parse & check if form has tagged media
Form saved as .xml file on the sdcard
Yes - media present, DownloadFormMediaTask executed
Download associated media files to device
Error
Control is passed onto the ODK's FormEntryActivity
Error / Failure
Control is passed onto the ODK's FormEntryActivity, but questions with media will not show associated media.

Figure 4.37. Activity diagram for survey retrieval module.

files) are present. If there are any media files to be downloaded then another task is executed to do the needful.

DownloadFormMediaTask: This is an *AsyncTask* responsible for fetching the all the media associated with the form.

Once the download process is completed and the form is fetched, control is then passed on to the ODK's *FormEntryActivity*. This is responsible for parsing the form.xml and rendering the user interface to answer the questions.

4.12 SURVEY SUBMISSION MODULE

After the user has answered all the questions required, then he needs to upload that data to the server to complete the survey.

The Figure 4.40 shows the sequence of data control flow that occurs during the survey submission process.

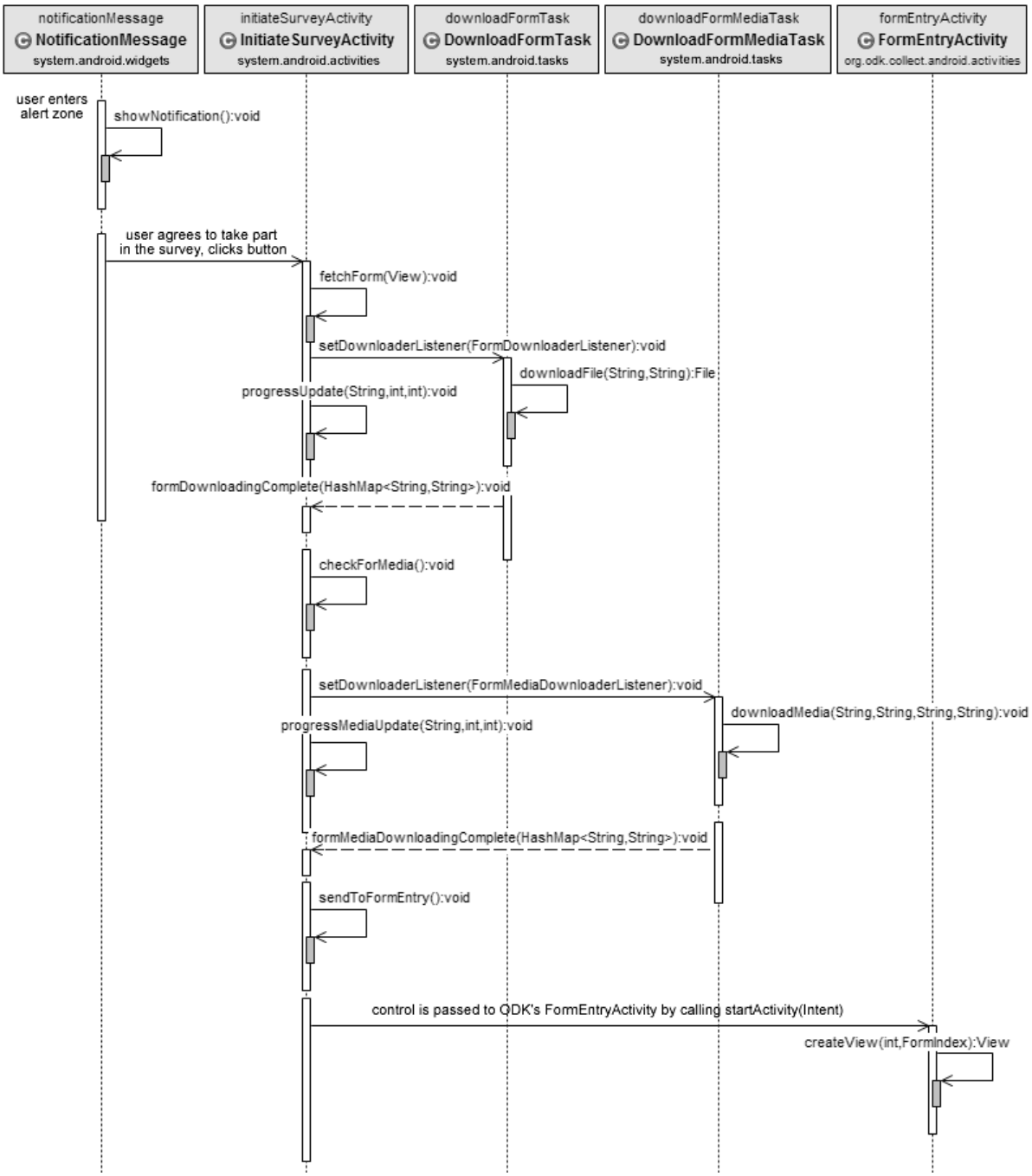


Figure 4.38. Sequence diagram explaining the survey retrieval process.

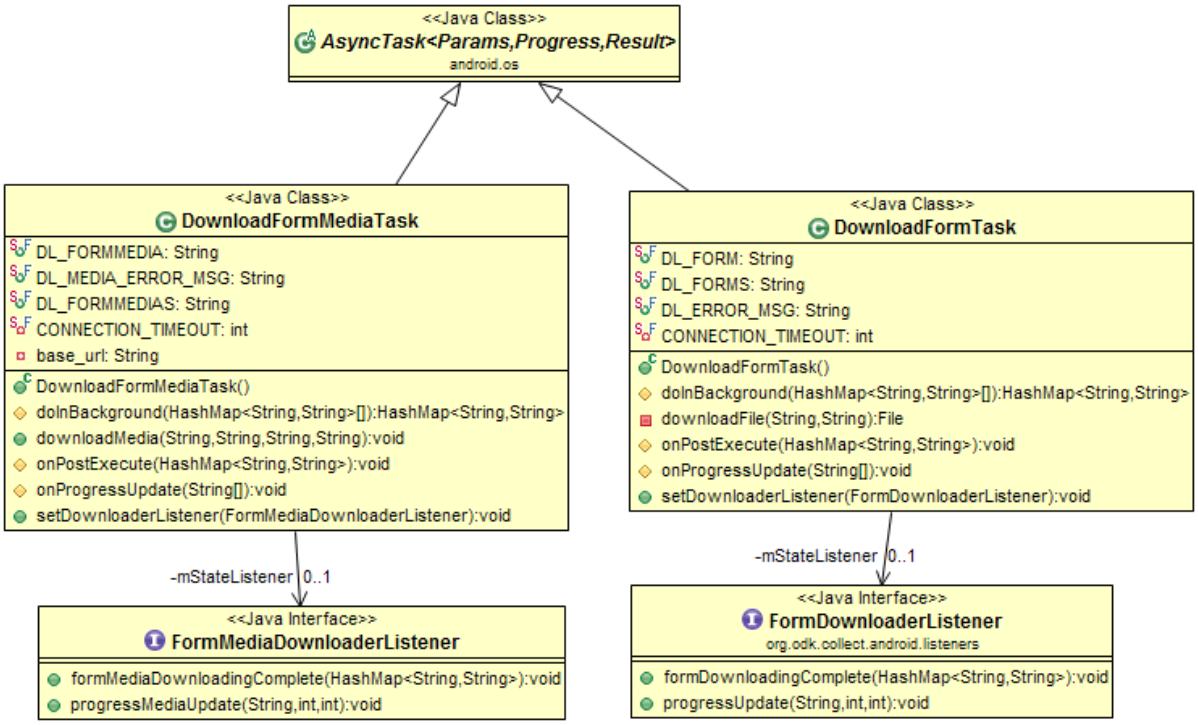


Figure 4.39. Class diagram for download tasks.

Once the user has reached the end of the survey, he needs to submit the data to the server. When the user clicks on the submit button an Android service is initiated to begin the submission process.

UploadInstanceService is the background service responsible for uploading the survey submission data completed by the user. It provides a visually rich feedback to let the user know what exactly is happening, as shown in the Figure 4.41.

4.13 UNREGISTER MODULE

The UnRegister Module allows the user to unregister himself from the server. As with the Registration module the server again contacts the Google's C2DM server to let it know that there won't be any C2DM messages that will be sent to that device and hence to delete it from their datastore. Once this is confirmed the user information is deleted from the server and the user no longer has communication with it.

The Figure 4.42 shows the sequence diagram for the mobile user unregistration process. After the successful completion of this process the Location service module takes control to remove all the location alerts currently active on the client device.

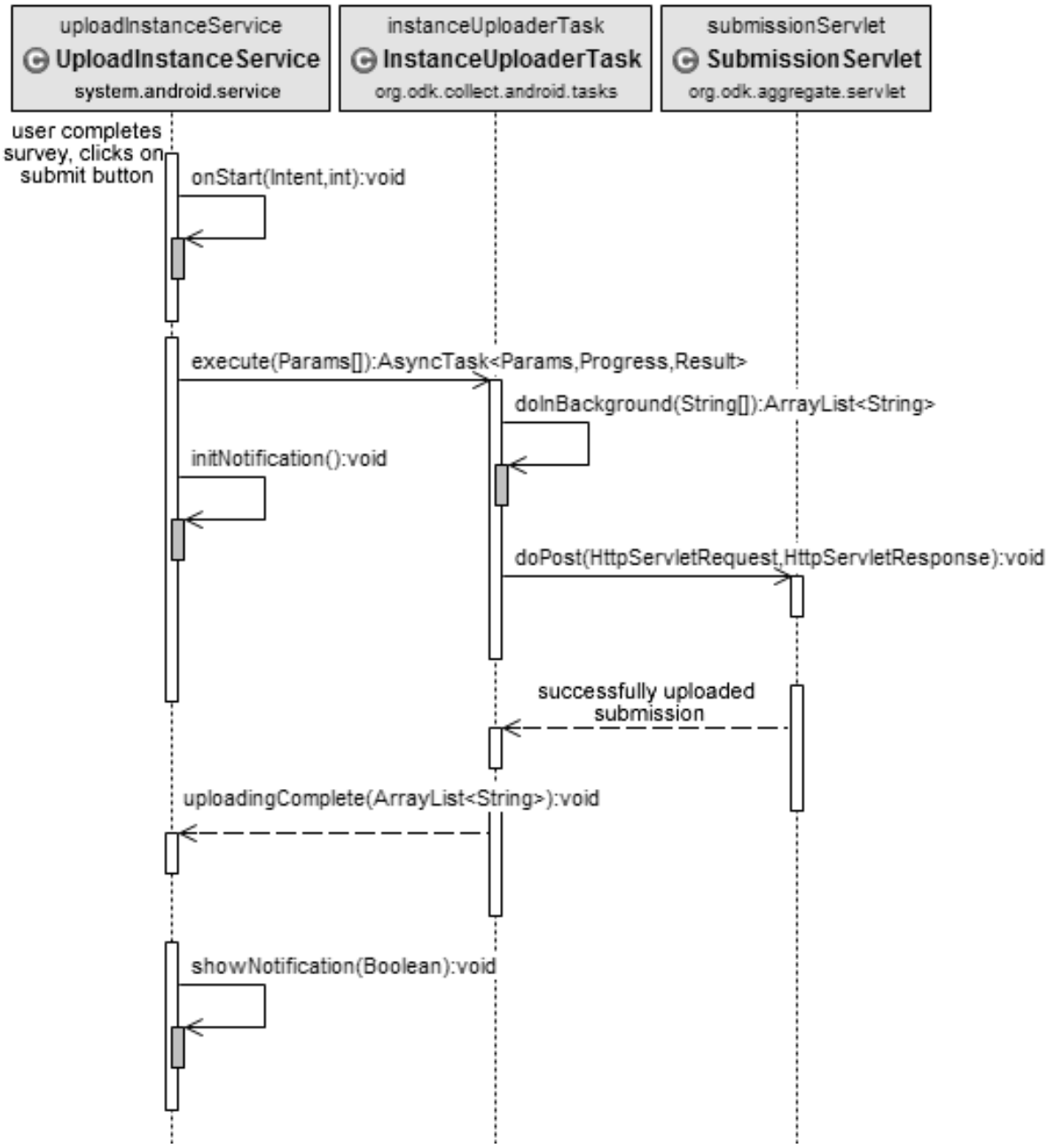


Figure 4.40. Sequence diagram explaining the survey submission process.

Figure 4.41. Survey uploading notification on mobile device.

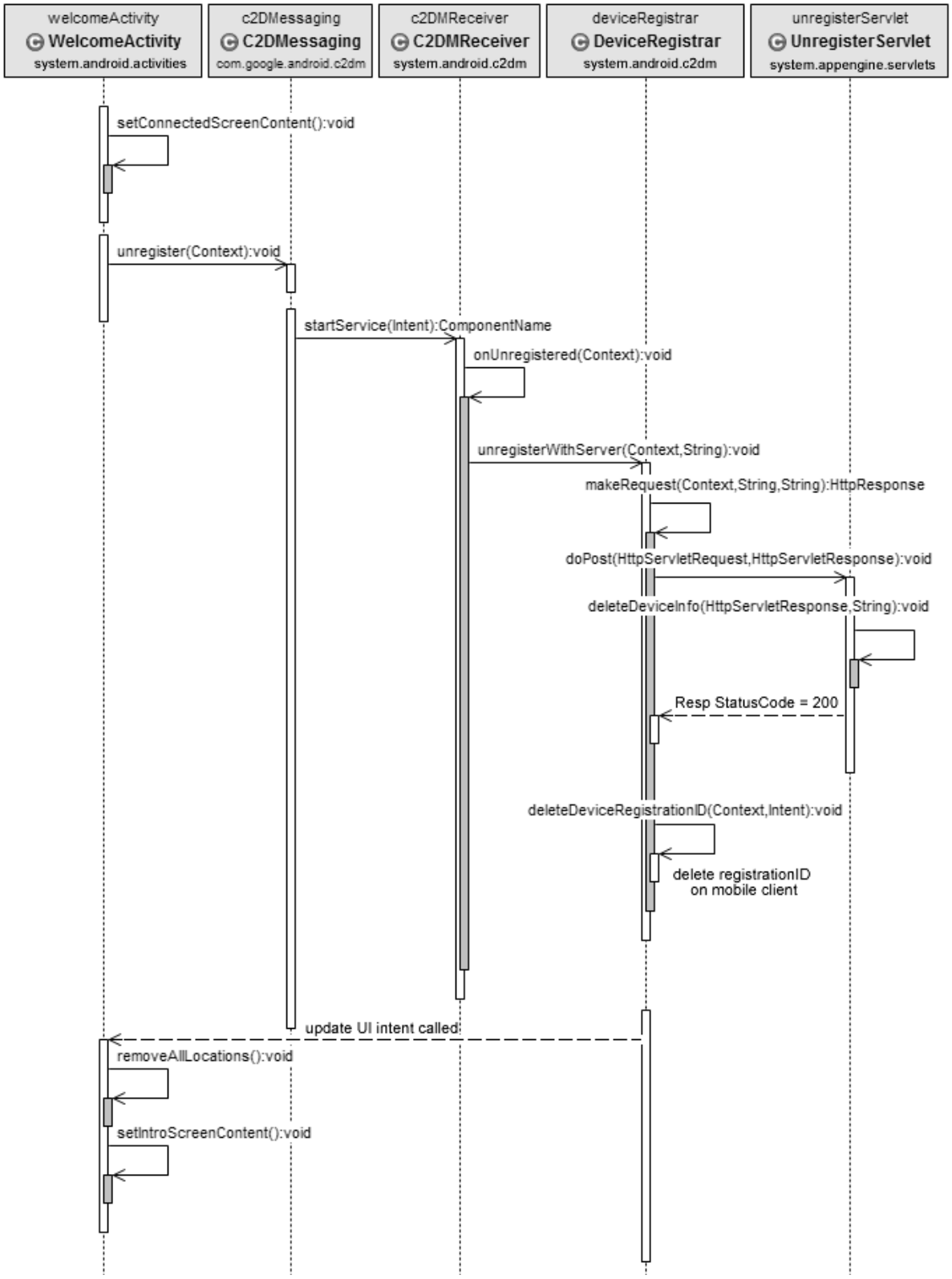


Figure 4.42. Sequence diagram explaining the user UnRegistration process.

CHAPTER 5

USABLE APPLICATIONS

5.1 SCENARIO ONE

This system can be used as an effective tool for organizations to assess the effectiveness of their customer service employees.

For the purpose of demonstration let's consider the example of a restaurant chain, say "Diner-X". The owner of this newly opened restaurant is very much keen on enhancing the customer service and to make sure that his employees are doing the best even in his absence. **The Proposed Solution:** During the weekly analysis he finds that on a particular day his sales declines every time. Hence to investigate more into this matter, he tailors the Momentary Assessment System for his business and publishes his service to the customer. He logs in to the system, creates a survey forms and tags it to the newly opened restaurant.

The Figure 5.1 shows a form created using the Momentary Assessment System for very purpose of demonstrating how this system can provide a better solution to the issue at hand.

The customer of the restaurant, who has already installed the mobile client application and has registered to the service, finds himself taking part in the survey as he is waiting for the order to be processed. He answers all the questions and uploads the data to the server. The owner of the restaurant during the next weekly analysis logs on to the system and starts to analyze all the uploaded data. Using this data he can determine exactly the time of the day when his customers were pleased or unhappy with the service. This would help him in finding out exactly who the employee was at that time and can take corrective measures.

The above explicated scenario is just one of the paradigms of how to make use of the Momentary Assessment System. The question which arises is what would make the user want to invest his /her time in taking the survey. The system can be tailored to include a discount coupon or other incentives upon successful completion of the survey as a means to motivate the user.

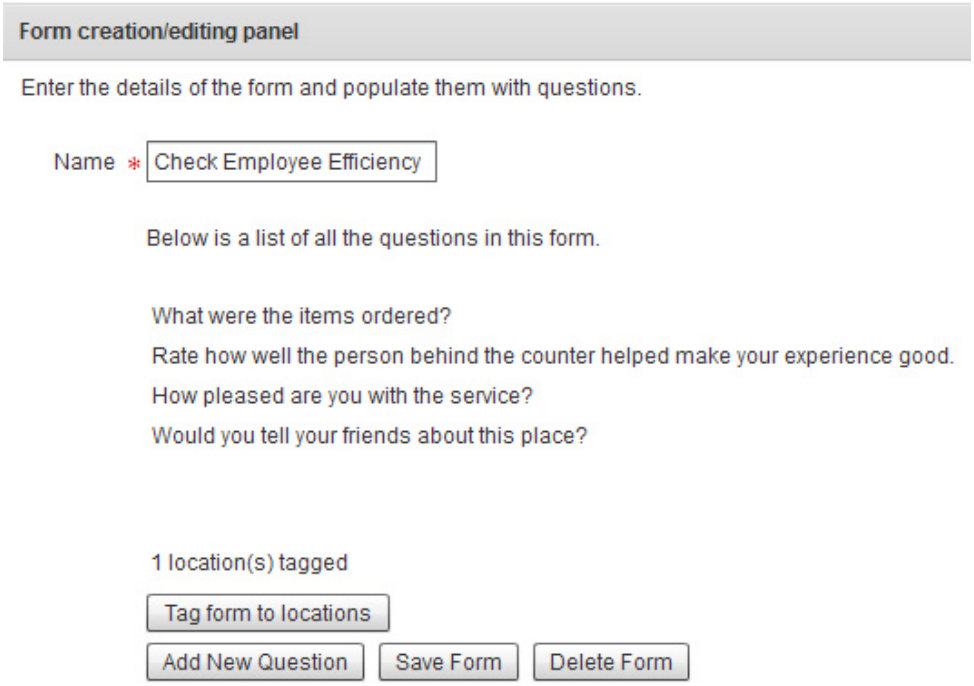

Form creation/editing panel
Enter the details of the form and populate them with questions.
Name *
Check Employee Efficiency
Below is a list of all the questions in this form.
What were the items ordered?
Rate how well the person behind the counter helped make your experience good.
How pleased are you with the service?
Would you tell your friends about this place?
1 location(s) tagged
Tag form to locations
Add New Question
Save Form
Delete Form

Figure 5.1. Application – Survey form to analyze employee effectiveness.

5.2 SCENARIO TWO

This system can be used to gather reviews of a place & build a web 2.0 site just like Yelp but with a few changes or advantages.

In the contemporary world, which is driven by web predominantly, one of the well known ways for businesses to advertise their service has been through social networking. This allows the users themselves to act as a medium to promote & propagate the business by means of recommendations, reviews and rating.

Yelp is one of the top providers of such service, which help people to connect with great local businesses. These businesses can be restaurants and shops; service businesses such as doctors, hotels, and cultural venues; and non-business locations such as schools, museums, parks, and churches etc. It allows users or Yelpers as they are referred to as, to write local reviews, rate business, upload images. This acts as a reference medium for other customers to check the credentials or the reputation of the place where they might be interested in going to.

The above said data does prove to be helpful but one cannot stop but wonder how accurate, fair and up to date it is. The drawbacks of this system are as follows:

- Lack of accuracy:
  Users can compose the reviews long after they have actually been to a business.
- Lack of restriction:
  Users can post reviews even if they haven't actually been to the business since there is no restriction or a gatekeeper to validate.
- Lack of user anonymity:
  Since a user is required to create a profile before he can review or rate, he might try to hold back on being completely honest to avoid being exposed.
  On November 3, 2009, a Yelp user in San Francisco [15] who had posted a criticizing review on a store had received angry messages and was once confronted by the owner of the store at his home. This eventually led to the store owner getting arrested by the police.

The Proposed Solution: The above stated drawbacks can be overcome by the use of a system which addresses all these issues. A social networking website can be built using the Momentary Assessment System proposed in this thesis following the Web 2.0 paradigm.

- The business owners can create a survey form or use one of the generic forms pertaining to the nature of the business and tag them to their store. This survey can consist of questions which would help users in judging the quality of the business.

The Figure 5.2 shows the survey form a fast food restaurant survey could have using the Momentary Assessment System. The user enters the business which is detected by the mobile client app and then prompts to take part in the survey or review process. The user answers all the questions and uploads the data to the server. This crowd sourced data can be used to build a more complex reputation system, whose representation could be a graph or a rating (say 5 star) system.

This leads to the possibility of creating a Web 2.0 application which is centered on a place of interest with data populated by relevant users.

5.3 SCENARIO THREE

This system can be used to trigger gathering of media/information on-demand during an emergency.

Twitter is a real-time information network which connects users to the latest information about whatever it is that the user is interested in. All the user has to do is follow

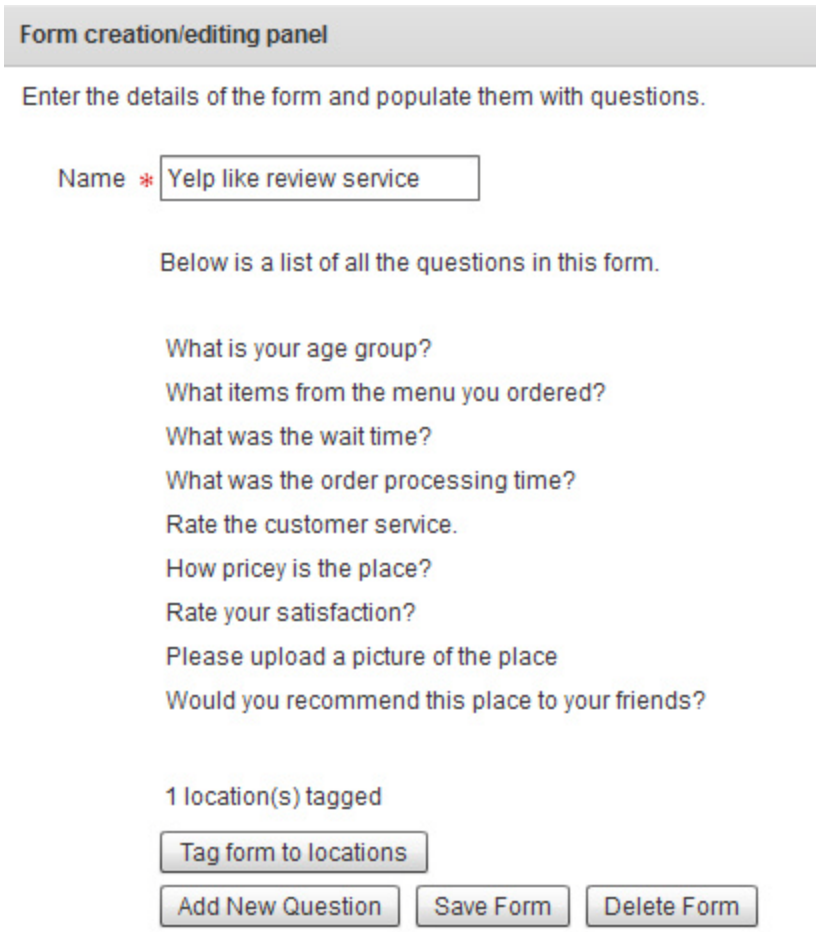

Form creation/editing panel
Enter the details of the form and populate them with questions.
Name *
Yelp like review service
Below is a list of all the questions in this form.
What is your age group?
What items from the menu you ordered?
What was the wait time?
What was the order processing time?
Rate the customer service.
How pricey is the place?
Rate your satisfaction?
Please upload a picture of the place
Would you recommend this place to your friends?
1 location(s) tagged
Tag form to locations
Add New Question
Save Form
Delete Form

Figure 5.2. Application – Survey form to create Yelp like review system.

public streams of the desired person and read the information they post on their profile. These small bursts of information are called Tweets.

This idea is user centered, but what if we could make it event-centered. Instead of the event/place posting to its profile, which make the feeds, what if the users who visit that event/place posted to the feeds. Then there would be a whole stream of user submitted information, be it text or audio or video or image, available to everyone.

Using the proposed Momentary Assessment System one can explicitly ask the connected users to take part in answering the survey and this could be posted live onto a web-stream. This ability to connect to all the devices is achieved by sending C2DM messages to all the connected devices or a subset of devices based on the nature of the request.

The Proposed Solution: For example, consider the recent scenario of earthquakes occurring in San Diego frequently. One can gather real time information from the mobile users in San Diego to assess the severity or even the knowledge of its occurrence. All that needs to be done is create a survey form, then filter all the devices by San Diego region and then send a C2DM message waking the mobile clients to fetch the form and get the users involved.

When the mobile users upload these data on to the server, the results can be used either as an analytical tool or as the above described live-stream/feeds.

The Figure 5.3 shows a survey form created using the Momentary Assessment System which could help prove to be a very useful tool in the context of the previously explained scenario.

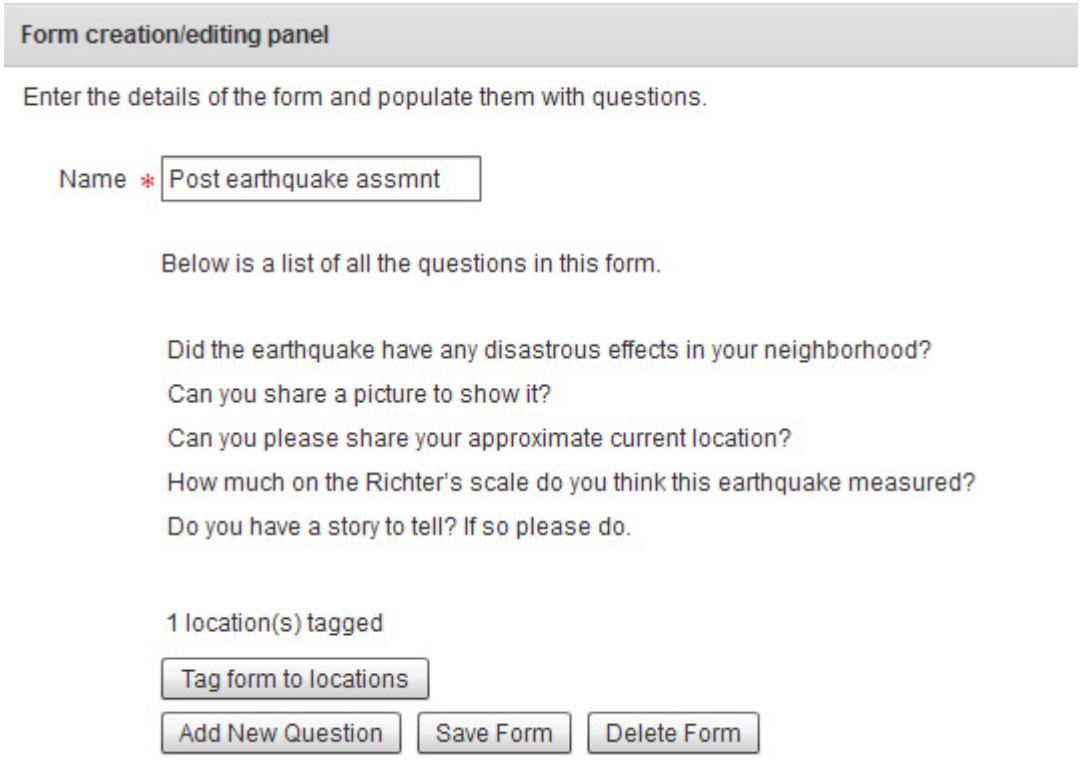

Form creation/editing panel
Enter the details of the form and populate them with questions.
Name *
Post earthquake assmnt
Below is a list of all the questions in this form.
Did the earthquake have any disastrous effects in your neighborhood?
Can you share a picture to show it?
Can you please share your approximate current location?
How much on the Richter's scale do you think this earthquake measured?
Do you have a story to tell? If so please do.
1 location(s) tagged
Tag form to locations
Add New Question
Save Form
Delete Form

Figure 5.3. Application – Survey form to gather data on demand.

CHAPTER 6

DEPLOYMENT GUIDE

This chapter is dedicated to educating developers on how to get started with development and installation of the application, both on the mobile client and also the server side.

6.1 SETTING UP DEVELOPMENT ENVIRONMENT

Pre-requisites for starting further development:

- Knowledge of programming languages Java, ActionScript
- Familiarity with Google Android framework for developing the mobile client application. For more development resources please refer http://developer.android.com/index.html
- Prior knowledge of C2DM framework. For more information please refer to http://code.google.com/android/c2dm/index.html
- Prior knowledge of building apps on Google App Engine. For more help please refer to http://code.google.com/appengine/
- Experience developing using Eclipse IDE.

Getting Started with Google App Engine project:

1. Sign up for a Google App Engine Account. For more information on this refer “How to create a new Google App Engine Application” section.
2. Download the Google App Engine SDK for Java
3. Download and install the Google plugin for Eclipse.
4. Download the application source code. Import the App Engine project into Eclipse as “Existing project”.

Getting Started with Adobe Flex project:

1. Download and install Adobe Flash Builder. There is a free trial available on the website http://www.adobe.com/products/flex/
2. Import the Flex project into the Flex builder as “Existing project”

Getting Started with Android project:

1. Download and install the Android SDK
2. Download and install the Android Development Tools (ADT) plugin for Eclipse.
3. Import the Android source project into Eclipse as “Existing project”.

After following all the above steps correctly, you have now successfully setup the development environment for starting further development on this thesis project. You can now start making changes to the source code and start testing them on your local machine.

6.2 DEPLOYING THE APPLICATION

This section explains in detail, step-by-step procedure on how to deploy the application onto a live audience. This section assumes that you have successfully set up all the development environment as explained in section. Please be advised to follow these steps in the exact same order to ensure that you do not run into any roadblocks.

6.2.1 Web Application

Here you are trained on how to successfully deploy the web application onto a Google App Engine server.

Pre-requisites:

1. You must have a working Google/Gmail account.
2. A new application must be created on the Google App Engine server with the same gmail account used to sign up, as described in step 1 of "Getting Started with Google App Engine project" in Section 6.1 Setting up development Environment.

Signing up for Android C2DM: In order for us to use the Android C2DM framework, we need to sign up an account for Android C2DM, which is currently in Google Labs developer beta (as of July 2011).

1. Go to http://code.google.com/android/c2dm/signup.html
2. Read and agree with the Android C2DM Terms of Service.
3. In the Sign up screen there are two sections, namely "Application Information" and "Contact Information"
   It is very important that you provide the same package name of the Android mobile client app as shown in Figure 6.1. Then fill in all the rest of the application details as shown in Figure 6.2

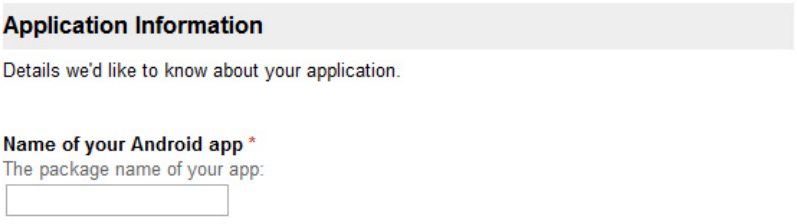

Application Information
Details we'd like to know about your application.
Name of your Android app *
The package name of your app:

Figure 6.1. Android C2DM signup application information.

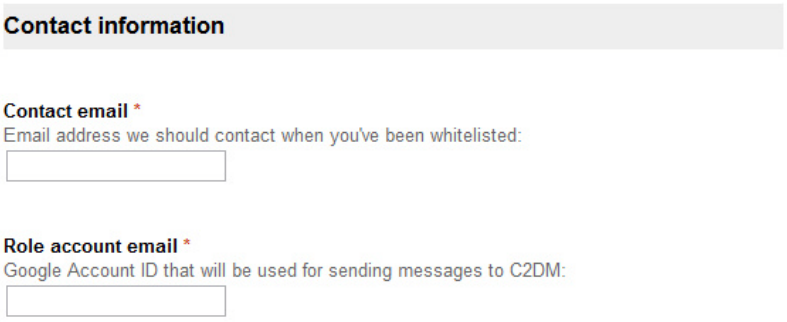

Contact information
Contact email *
Email address we should contact when you've been whitelisted:
Role account email *
Google Account ID that will be used for sending messages to C2DM:

Figure 6.2. Android C2DM signup contact information.

In contact information section's "Role account email" is the Google account that will be responsible for sending messages to C2DM. Preferably it should be the same account that was used in "How to create a new Google App Engine Application" section. Then click "Submit" button.

4. The signup process may take up to 24hrs for approval. You will receive a confirmation email once it has been approved.
   Note: You will not be able to send C2DM messages until the confirmation email has been received.

Creating a new Google App Engine application:

1. Login to https://appengine.google.com/ using your gmail account.
2. In the next screen as shown in the Figure 6.3, click on the "Create Application"

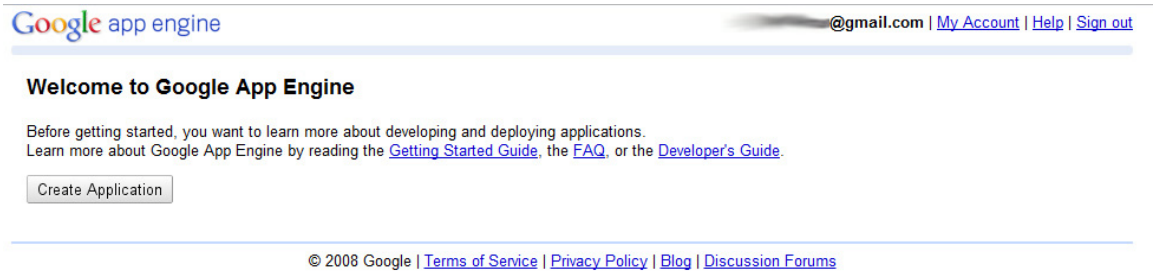

Google app engine
[redacted]@gmail.com | My Account | Help | Sign out
Welcome to Google App Engine
Before getting started, you want to learn more about developing and deploying applications.
Learn more about Google App Engine by reading the Getting Started Guide, the FAQ, or the Developer's Guide.
Create Application
© 2008 Google | Terms of Service | Privacy Policy | Blog | Discussion Forums

Figure 6.3. Google app engine new application welcome screen.

3. In the "Create an Application" screen (Figure 6.4) enter the "Application Identifier" and the "Application Title". Then click on "Create Application" button to complete the process.
   Note: The identifier used will result in the http://<app-identifier>.appspot.com url for your application.

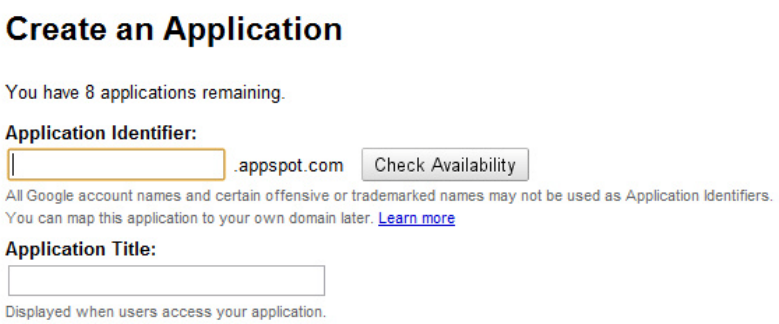

Create an Application
You have 8 applications remaining.
Application Identifier:
.appspot.com
Check Availability
All Google account names and certain offensive or trademarked names may not be used as Application Identifiers.
You can map this application to your own domain later. Learn more
Application Title:
Displayed when users access your application.

Figure 6.4. Google app engine create an application.

Exporting a Flex project: Any changes made to the Flex project needs to be first exported and copied to the App Engine's "war" folder. Before we export we need to make a few changes in the code.

1. Signup for Google Maps API here http://code.google.com/apis/maps/signup.html and obtain a key.
2. Open "Constants.as" file in package "system.flex.thesis.utils" found in the source code and change the constant "GOOGLE_MAPS_KEY" to use the obtained key.
3. In the same "Constants.as" file, change the constant "URI_BASE" to use the new application url *http://<app-identifier>.appspot.com/ as* explained in section "How to create a new Google App Engine Application".
   Note: Please look for the already existing string urls and make sure that it adheres to the same format.
4. In the package explorer, right-click on the project folder and select "Export"
5. Export dialog pops up as shown in the Figure 6.5

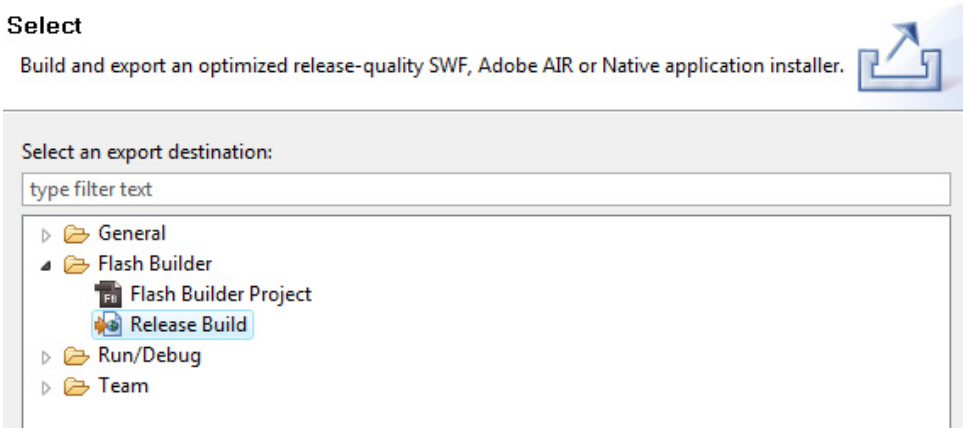


Figure 6.5. Export flex application.

6. Select the “Release build” option and click “Next”.
7. The next screen prompts us to select the project and the mxml application that needs to be built. The project name included in the source is “MASFMD” and the application is “index.mxml”. The default release build directory is /bin-release, which can be changed as per requirement. Then click “Finish”.
8. Now the release files are available in the directory specified in the previous step. As per the above steps “index.html” and “index.swf” are the two files that are required for deploying on the App Engine server.

Obtaining User-Auth token: Before we actually deploy the Google App engine we need to first fetch the User-Auth token, which is used by the Android C2DM framework to send messages to devices that are connected. This procedure is to be carried out just one time, once it’s in the system everything else is automated.

1. We need to run the following curl script command to fetch the auth-token.
   Note: If you are running on a Mac no setup is required, just run the following command in the terminal. But if you are running on a Windows PC, then please ensure that curl commands can be run on the system.

   curl https://www.google.com/accounts/ClientLogin -d Email=<appengine-role-account>@gmail.com -d "Passwd=<account-password>" -d accountType=GOOGLE -d source=https://android.clients.google.com/c2dm/send -d service=ac2dm

   Please ensure that this is one command copied exactly as shown and without any line breaks. In the “Email=” enter the one specified in “Role account email” text box as described in “How to sign up for C2DM” section as the parameter. In the “Passwrd=”, please enter the password of the same account.

2. If the command runs successfully then you will get an output in the terminal which consists of 3 variables namely SID, LSID and Auth. Please copy the complete alpha numeric string after the “Auth=” and save it. This is the user Auth token which will be used.

Deploying to the Google App Engine:

1. Ensure that the two files exported from step#5 of “How to export a Flex project” is copied to the “project-name/war/” folder.
   Note: If you want to change the welcome file, please refer the “web.xml” file in “project-name/war/web-inf” folder.
2. We have to include the user auth-token obtained after completion of section “How to obtain User-Auth token” in the app engine project. Open the “C2DMConfigLoader.java” in “com.google.android.c2dm.server” package of the source code. Change the string constant “USER_AUTH_TOKEN” to the newly obtained user-auth token string.

3. In the package explorer, right-click on the project folder and navigate to “Google” and click on “Deploy to App Engine” as shown in Figure 6.6
Note: If you do not see the “Google” option, then install the Google plugin for Eclipse

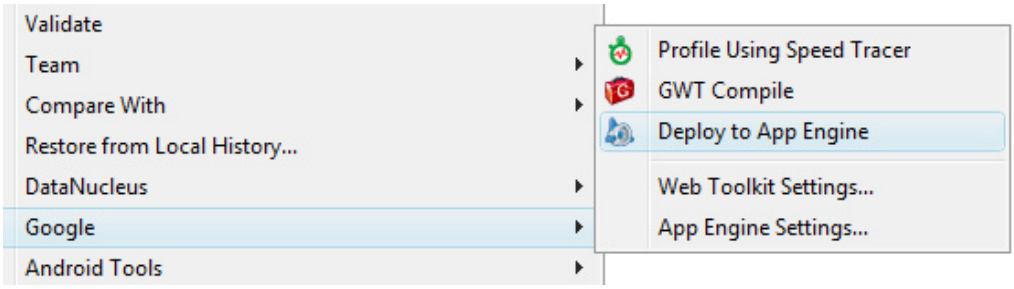


Figure 6.6. How to deploy to Google app engine.

4. In the deploy popup screen as shown in Figure 6.7, enter the Google account that was used in “How to create a new Google App Engine Application” section and also the password.

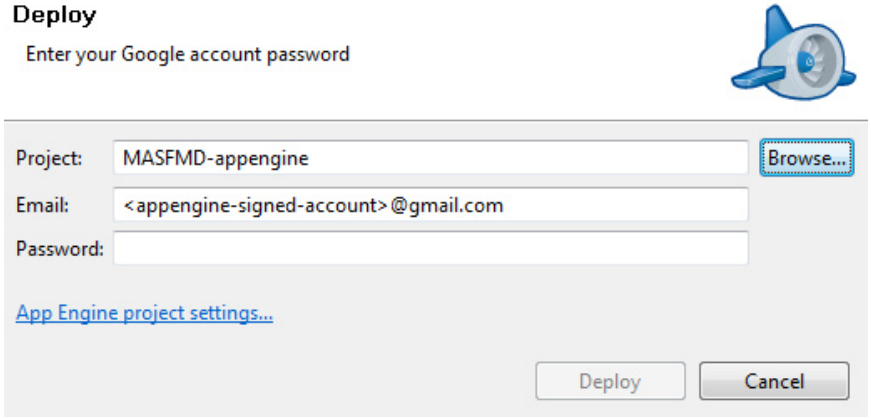


Figure 6.7. App engine deploy signin.

5. Before you click on “Deploy” button, click the “App Engine project settings” link to check the application settings.

   As shown in the Figure 6.8, choose the appropriate App Engine SDK. In the “Application ID” text box enter the identifier used in “How to create a new Google App Engine Application” section. Enter the desired version number and click Ok.

6. Now click on the “Deploy” button to start the deployment process.
Note: Look at console window on Eclipse to check the progress.

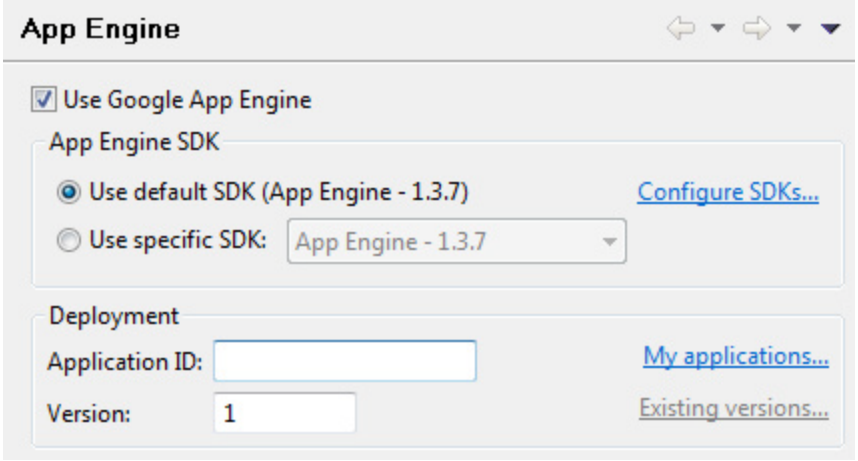


Figure 6.8. App engine project settings.

6.2.2 Mobile Client Application

Here you are trained on how to successfully build the Android application, sign it, after which the resulting .apk file can be forwarded to the user or uploaded onto the Android Market.

Build an Android application:

1. Change the url of the application to the already created new App Engine application as described in “How to create a new Google App Engine Application” section. Open the “AppEngineClient.java” in “system.android.thesis.c2dm” package of the source code. Change the string constant "BASE_URL” to http://<app-identifier>.appspot.com
2. Change the C2DM messages senderID to the one specified in “Role account email” text box as described in “How to sign up for C2DM” section. Open the “DeviceRegistrar.java” in “system.android.thesis.c2dm” package of the source code. To do this change the string constant “SENDER_ID” to <c2dm-role-account>@gmail.com
3. In the package explorer, right-click on the project folder and navigate to “Android Tools” and click on “Export Signed Application Package” as shown in Figure 6.9 *Note: If you do not see the “Android Tools” option, then install the ADT plugin for Eclipse.*
4. In the popup screen that appears provide the Keystore (<key-name>.keystore> to sign the application with. If you do not have one then there is an option to create a new keystore within Eclipse. For more information refer http://developer.android.com/guide/publishing/app-signing.html

5. Successful completion of step#4 will result in creation of an apk file, which can be either published to Android market or forwarded to the users directly to install. If you

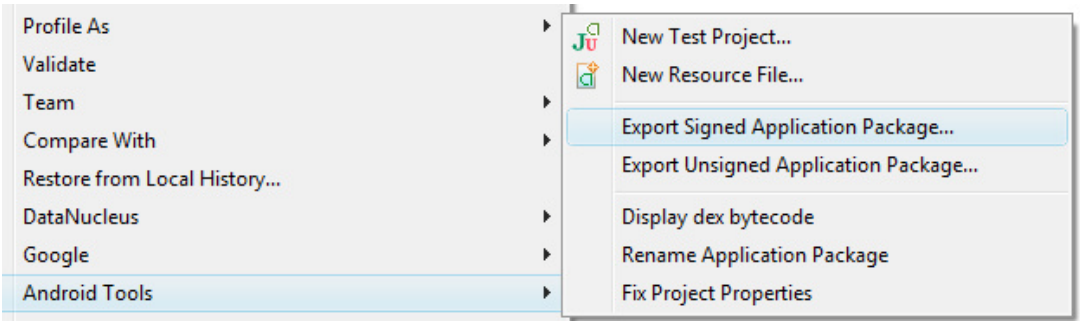


Figure 6.9. Export signed Android application package.

choose the latter then please refer the section “How to install Android Application on the device”.

CHAPTER 7

USER MANUAL

Provide a user Admin access: The owner of the application by default is the administrator.

1. Open the url https://appengine.google.com/ on your browser and login.
2. Click on the appropriate application to get more details.
3. On the left hand side, under the section “Administration” please click on the “Permissions” link.
4. This will shown a page where you can edit current access roles or add new users to collaborate with your application as shown in Figure 7.1. There are 3 different roles which are provided by the Google App Engine server namely Owner, Developer, Viewer. For this application all the 3 roles gives admin access to the momentary assessment system. Once you enter the desired user email address, choose a role and click on the “Invite User” button. This will send a mail to that user asking him to accept this request. Once he agrees to the terms then he can login to the momentary assessment system as an admin.

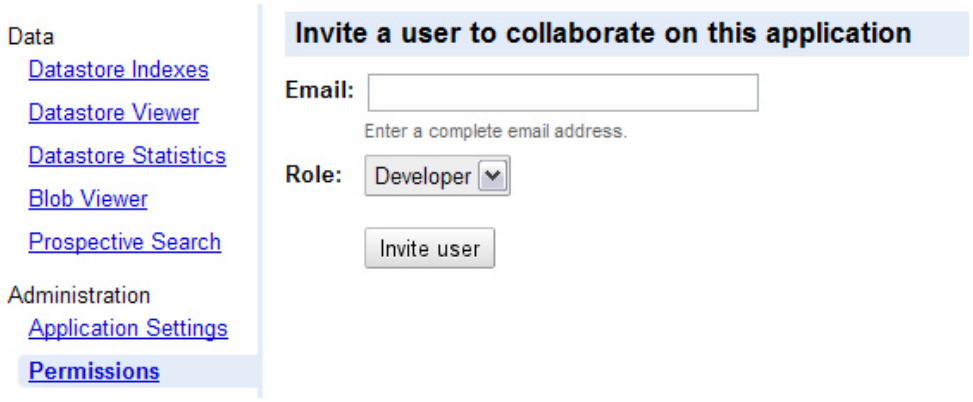


Figure 7.1. Provide admin access to user.

Provide a user access to the system: The owner of the application by default is the administrator. All administrators have access to the system by default even if their email address isn’t explicitly declared in the system. If any general user tries to login to the system, he will not have access to any of the modules and will be shown an “Access Required” message. Hence the administrator needs to explicitly provide access to each user.

1. Open the url http://momentaryassessment.appspot.com/ on your browser and login to the system.

2. Click on the “Admin Panel” link button on the top right hand corner of the page, to the left of the logout button.

   The popup panel shown in Figure 7.2 is called the Admin Panel. Here it consists of a list of all the already entered email addresses and an option to edit, delete or add new users on the right hand side as shown in the Figure 7.2.

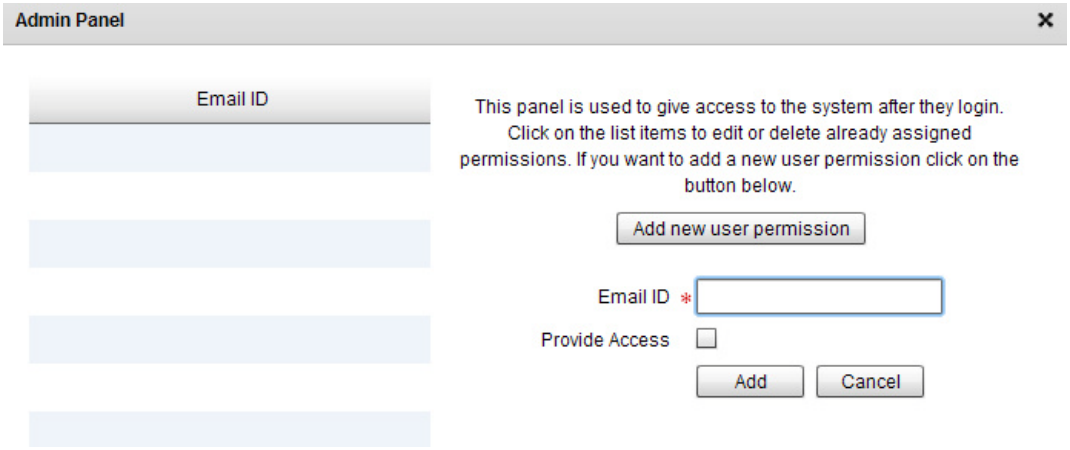

Admin Panel
Email ID
This panel is used to give access to the system after they login. Click on the list items to edit or delete already assigned permissions. If you want to add a new user permission click on the button below.
Add new user permission
Email ID *
Provide Access
Add
Cancel

Figure 7.2. Admin panel to provide system access to users.

3. Click on the “Add new user permission” button. Enter the desired email address, check the “Provide Access” checkbox and click Add.
   Note: If the “Provide Access” checkbox is not checked then the user email address will be added to the system but he will not have access to the system yet.
4. If you want to edit existing permissions, click on the respective email address on the list and edit it.

Creating a new survey:

1. Open the url http://momentaryassessment.appspot.com/ on your browser.
2. Click on the “Form Builder” tab at the top to bring up the Form/Survey creation and editing panel as shown in the Figure 7.3.

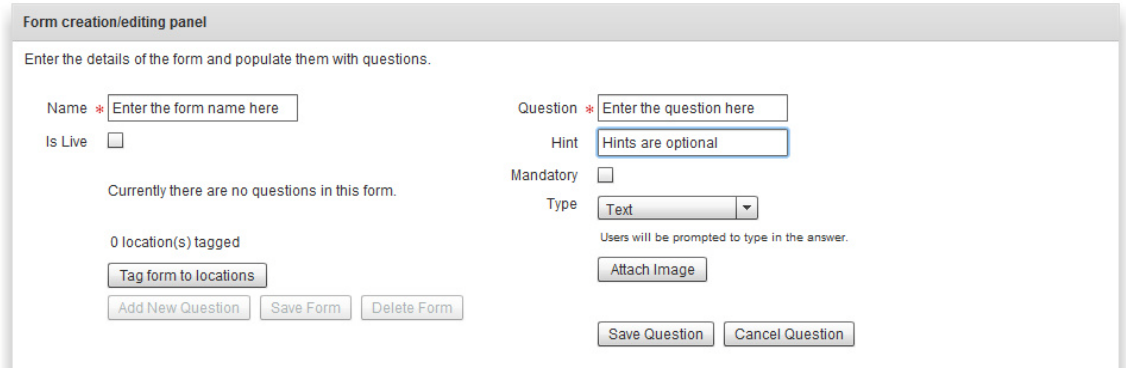

Form creation/editing panel
Enter the details of the form and populate them with questions.
Name * Enter the form name here
Is Live
Currently there are no questions in this form.
0 location(s) tagged
Tag form to locations
Add New Question
Save Form
Delete Form
Question * Enter the question here
Hint Hints are optional
Mandatory
Type Text
Users will be prompted to type in the answer.
Attach Image
Save Question
Cancel Question

Figure 7.3. Form creation/builder panel.

3. Enter all the details and start adding questions to the survey form by clicking on the "Add New Question" button which brings up the question creator panel on the left.
4. Enter the question text and hints (if required). You can also check the "Mandatory" option if you want the mobile user to mandatorily answer the question. The field "Type" refers to the types of questions currently supported by the system. Click on the dropdown box to bring them up and select the desired option. *Note: A brief help text is shown below the field corresponding to the selected question type.*
   Image can also be tagged to each of these questions by clicking on the "Attach Image" button and choosing the desired image file. Then click "Save Question" button to add to the list.
5. To tag location to the survey form, click on the "Tag form to locations" button. This brings up a pop up with a list of all the available locations in the system.
   Note: Refer the section "How to create a new location" for more information.
6. There is form validation included to help you. Once all the questions are entered, click on the "Save Form" button. An appropriate progress dialog popup is shown during the saving process and then the page is redirected to the "Form Dashboard" panel.

Creating a new location: The Location dashboard panel helps the administrative users to add locations to the system. It uses Google Maps to render locations and supports all the standard Google Maps features except Street View.

1. In the main menu, click on the "Location Dashboard" tab to bring up the panel as shown in the Figure 7.4.

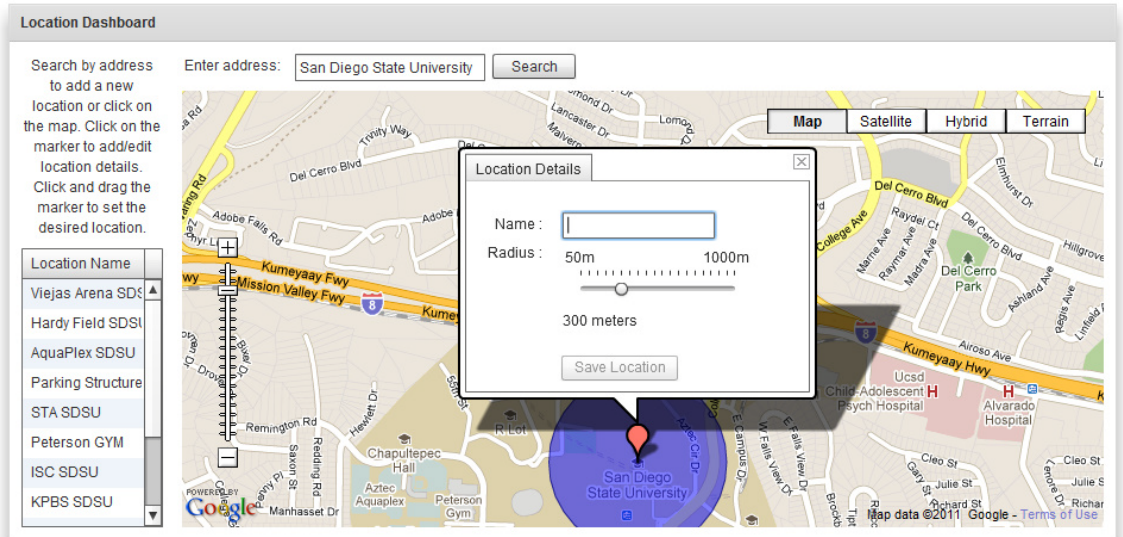


Figure 7.4. Location dashboard view.

2. Follow the descriptive text on the left side of the panel to choose a location.
3. Once the location is chosen a marker is drawn on the map surrounded by a blue-colored circular region and a popup appears with the tab “Location Details” selected to enter details of the location.
4. Enter the name of the location and choose the alert radius. This alert radius corresponds to the region within which a form survey request must be triggered to the mobile user. As you change the value of the radius on the horizontal scroll bar, a visual feedback is provided helping you to easily determine the required radius. *Note: You can pan around the map with the marker in place also. If you close the popup for the marker, then it can be viewed again by clicking on the marker itself.*
5. Then click on the “Save Location” button. A progress dialog popup is shown during the saving process and then the saved location is populated in the list on the left side of the panel.

Note: You can also click on the already existing locations (present on the left side of the panel) to view their details. In this mode, there is a new tab which is added to the marker popup called “Tagged Forms”. This tab shows the form currently tagged to that location.

View Survey submissions:

1. In the main menu, click on the “Form Dashboard” tab to bring up the dashboard.
2. The left side of the panel is a list of the forms available in the system. Click on the desired form to bring up the details as shown in Figure 7.5.

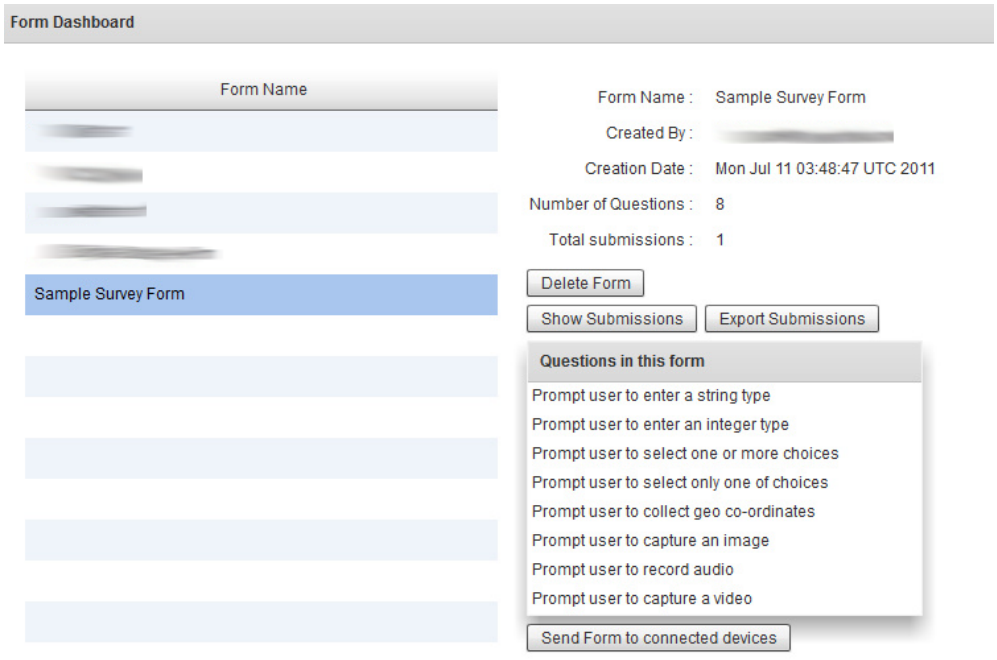

Form Dashboard
Form Name
Sample Survey Form
Form Name : Sample Survey Form
Created By :
Creation Date : Mon Jul 11 03:48:47 UTC 2011
Number of Questions : 8
Total submissions : 1
Delete Form
Show Submissions
Export Submissions
Questions in this form
Prompt user to enter a string type
Prompt user to enter an integer type
Prompt user to select one or more choices
Prompt user to select only one of choices
Prompt user to collect geo co-ordinates
Prompt user to capture an image
Prompt user to record audio
Prompt user to capture a video
Send Form to connected devices

Figure 7.5. Form dashboard showing form details.

3. Now the right side of the panel shows the details of the form selected. Click on the "Show Submissions" button.
   Note: This button is disabled if there are no submissions yet for the survey form.
4. This brings up the submission viewer panel as shown in the Figure 7.6. It has a question carousel at the top to navigate through the questions available in the survey form.

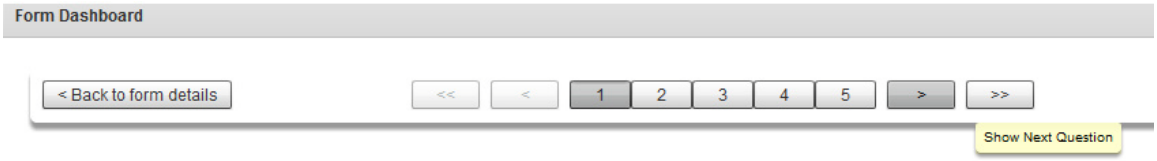


Figure 7.6. Submission viewer showing question carousel.

You can cycle through the questions be clicking on the two buttons on either end of the carousel. You can also jump to particular question by clicking on the desired question number. Also at the bottom end of this panel there is also details about the current opened survey form (not shown in the Figure 7.6). To the left side it shows the name of the form and on the right side it shows the total submissions available for this form.
Tip: Move mouse cursor over each of the buttons to get the tool tip as shown in the Figure 7.6. If you hover over the question number, it previews the actual question title.

Export Survey submission data:

1. In the main menu, click on the "Form Dashboard" tab to bring up the dashboard.
2. The left side of the panel is a list of the forms available in the system. Click on the desired form to bring up the details.
3. Now the right side of the panel shows the details of the form selected. Click on the "Export Submissions" button.
4. A popup is shown, asking the directory where the export data file needs to be saved. Please choose the desired location and click ok. This will save the exported data in an excel sheet.

Send forms directly to connected devices: The Device Monitoring Panel is used to send forms directly to connected devices. It has a list of all the devices connected with information on about those devices too.

Note: The information showed currently in this thesis project is just for demonstration purposes only. The system does not save any sensitive information like the phone number nor shares any shown information.

1. In the application's main menu, click on the "Form Dashboard" tab to bring up the dashboard.

2. Click on the desired survey form from the list shown on the left side to view more details about it, as shown in step#2 of “How to view Survey submissions” section.
3. Click on the “Send Form to connected devices” button to go to the Device Monitoring Panel as shown in the Figure 7.7.

Figure 7.7. Device monitoring panel.

4. To send form to a particular device, click on it. When the message is successfully sent an appropriate popup is shown.
   To send form to all the devices, click on the “Send to all Devices” button.

Installing an Android Application on the device: This section assumes that the Android application is not published on the Android Market. This type of installation is referred to installing from Unknown Source. To do so first we have to setup the device to accept installation of non-market applications, as described in the steps below.

1. Go to the Home screen of the device.
2. Press the hardware MENU-KEY to bring up the menu.
3. Click on Settings.
4. Click on Applications.
5. Check the “Unknown sources” options, as shown in the Figure 7.8.

Now your device is configured to install applications of Unknown source. Open the browser and type in the url provided to download the .apk file on to your device. Then follow the on screen instructions to complete the installation process.

Instructions to use the Android application - User Registration: This section assumes that the mobile user has already installed the application on the device. For more information refer “How to install Android Application on the device” section.

1. Open the Android apps drawer from the home screen and the click on the application name. In this case it’s called “MASFMD”.
2. You will be greeted by the welcome screen, then click on the “Next” button.

Figure 7.8. Settings to install Android application from unknown sources.

3. This brings up the registration screen which contacts the server to establish connection. Choose the Google account you want to use to register with and then click next. The device
4. This brings up the registration screen which contacts the server to establish connection. Choose the Google account you want to use to register with and then click next. The device is now connecting to the server and an appropriate "Connecting" progress bar is displayed. The Figure 7.9 shows the stages of this registration process.
   Note: Please make sure that data connection is available.

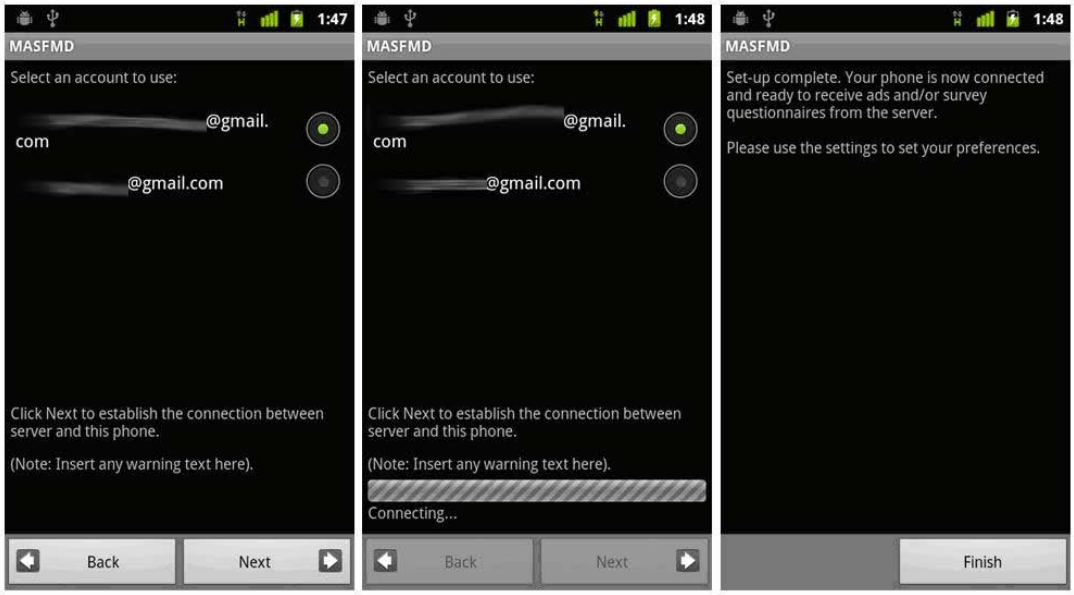


Figure 7.9. Android application showing the sequence of the registration process.

5. Once the register is completed you are navigated to a new screen which confirms the same. Now your device is now connected to the service and ready to receive survey forms.

User Un-Register/Disconnect: This section assumes that the mobile user has already registered the device on the server. For more information refer “How to Register” section.

1. Open the Android apps drawer from the home screen and the click on the application name. In this case it’s called “MASFMD”.
2. Now the connected screen is show. Click on the “Disconnect” button to unregister as shown in Figure 7.10. Once this button is clicked an appropriate progress bar is shown to indicate that the process has begun.
3. Once the device has been un-registered from the server, the welcome screen is shown to the user.

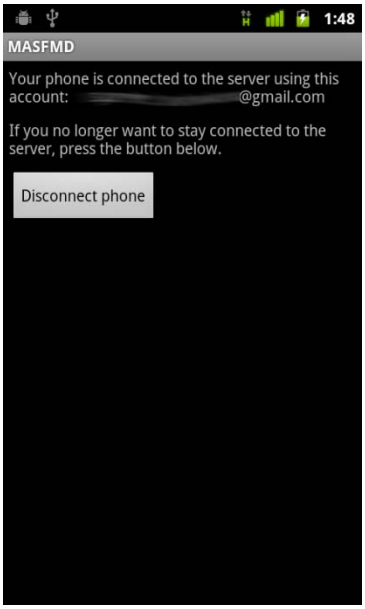

1:48
MASFMD
Your phone is connected to the server using this account:
@gmail.com
If you no longer want to stay connected to the server, press the button below.
Disconnect phone

Figure 7.10. Android application how to disconnect/unregister.

CHAPTER 8

FUTURE ENCHANCEMENTS

The Momentary assessment system described above has the basic functionality to prove the concept. Taking into consideration all the technologies used, there is lots of scope for improvement.

Currently the survey submissions are uploaded anonymously to the server. The application can be further developed to include mobile user profiles to save a copy of that on the cloud, keep track of the places he has been or those he might be interested in. It could also help build a social network around the profile like adding avatars and linking to user's facebook profile.

The idea of momentary assessment system involves around locations which can take the form of stores, restaurants, localities or other businesses. This could mean the further development of the location module to include store names, branches associated with them instead of plain locations (as implemented currently). Combining this feature and the admin login, we can have a user associated with each business which would help maintain the location specific activities.

On the mobile client side of the application we can add functionality to let the end users choose the category and or particular store, restaurants etc that he would be interested in.

Furthermore, the addition of an incentive system to encourage users to get more active on submitting reviews could prove helpful and a means to gather more traffic to the business. This incentive system can be either discount coupon or keeping scores which could be later used to gift users.

CHAPTER 9

CONCLUSION

The main purpose of this thesis was to introduce an approach to gather information from the mobile client users driven by an event or location. It also provides a user interface to analyze the data submitted by the mobile client users in an organized, graphical manner. The possible applications for this system have been described in detail, with sample survey form templates in Chapter 5 – Usable Applications. It also shows how to build a system keeping in mind the existing standards so it can be compatible with other systems which follow the same norms. This project also introduces Google's C2DM framework and how to take advantage of its functionalities.

The Android application was tested on devices which were equipped with Android 2.2 (a.k.a Froyo) and above. The screenshots taken to illustrate it were taken on Nexus One device running Android 2.3 (a.k.a Gingerbread). The web client was tested on Google Chrome, Mozilla Firefox browsers with Adobe Flash Player v10.3.181.26 installed on it.

REFERENCES

[1] WIKIPEDIA, *Electronic commerce*. Wikipedia, http://en.wikipedia.org/wiki/Electronic_commerce, accessed July 2011, 2011.

[2] WIKIPEDIA, *Smartphone*. Wikipedia, http://en.wikipedia.org/wiki/Smartphone, accessed July 2011, 2011.

[3] J. HOWE, *Crowdsourcing: A definition*. Crowdsourcing, http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html, accessed July 2011, 2006.

[4] M. VUKOVIC, S. KUMARA, AND O. GREENSHPAN, eds., *Proceedings from UbiComp '10*, Copenhagen, Denmark, 2010, UbiComp.

[5] D. BARTH, *The bright side of sitting in traffic: Crowdsourcing road congestion data*. Google Blog, http://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html, accessed July 2011, 2009.

[6] WIKIPEDIA, *Social networking service*. Wikipedia, http://en.wikipedia.org/wiki/Social_networking_service, accessed July 2011, 2011.

[7] YELP, *What is Yelp*? Yelp, http://www.yelp.com/faq#what_is_yelp, accessed July 2011, 2011.

[8] A. K. DEY, G. D. ABOWD, AND D. SALBER, *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*. Human-Computer Interaction, 16 (2001), pp. 97-166.

[9] H. I. MYRHAUG, N. WHITEHEAD, A. GOKER, T. E. FAEGRI, AND T. C. LECH, eds., *Proceedings from the Second European Symposium, EUSAI 2004*, Eindhoven, The Netherlands, 2004, EUSAI.

[10] R. LORRAINE COLLINS, T. B. KASHDAN, AND G. GOLLNISCH. *The feasibility of using cellular phones to collect ecological momentary assessment data: Application to alcohol consumption*, Exp. Clin. Psychopharmacol., 11 (2003), pp. 73-78.

[11] S. SHIFFMAN AND A. A. STONE, *Introduction to the special section: Ecological momentary assessment in health psychology*. Health Psychol., 17 (1998), pp. 3-5.

[12] CISCO, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010–2015*. Cisco, San Jose, California, 2011.

[13] CISCO, *Enterprise Mobility 4.1 Design Guide*, Cisco, San Jose, California, 2009, Ch. 13, pp. 1-9.

[14] D. GHOSH, *Building push applications for Android. Live session Google I/O 2010*. Google, http://www.google.com/events/io/2010/sessions/push-applications-android.html, accessed July 2011, 2010.

[15] WIKIPEDIA, *Yelp. Inc*. Wikipedia, http://en.wikipedia.org/wiki/Yelp,_Inc., accessed July 2011, 2011.

APPENDIX A

ABBREVIATIONS

ABBREVIATIONS

1.	**PDA**	Personal Digital Assistant
2.	**GPS**	Global Positioning System
3.	**PM**	Paper-and-pencil self-Monitoring
4.	**EMA**	Ecological Momentary Assessment
5.	**3G**	3rd Generation
6.	**VNI**	Visual Networking Index
7.	**LBS**	Location Based Services
8.	**RTLS**	Real Time Location Systems
9.	**SDK**	Software Development Kit
10.	**JDO**	Java Data Object
12.	**JPA**	Java Persistence API
13.	**API**	Application Programming Interface
14.	**JVM**	Java Virtual Machine
15.	**JSP**	Java Server Pages
16.	**WAR**	Web application Archive
17.	**JNI**	Java Native Interface
18.	**OHA**	Open Handset Alliance
19.	**GNU**	GNU's Not Unix!
20.	**VM**	Virtual Machine
21.	**RIA**	Rich Internet Application
22.	**ODK**	Open Data Kit
23.	**TCP**	Transmission Control Protocol
24.	**mA**	milliAmps
25.	**C2DM**	Cloud to Device Messaging
26.	**GAE**	Google App Engine

APPENDIX B

USEFUL LINKS

USEFUL LINKS

Open Data Kit Group

- http://www.opendatakit.org/

Open Data Kit's open source project home

- http://code.google.com/p/opendatakit/

My personal website

- http://www.supreeth.net/thesis.html

Master's Thesis Project

- http://momentaryassessment.appspot.com/

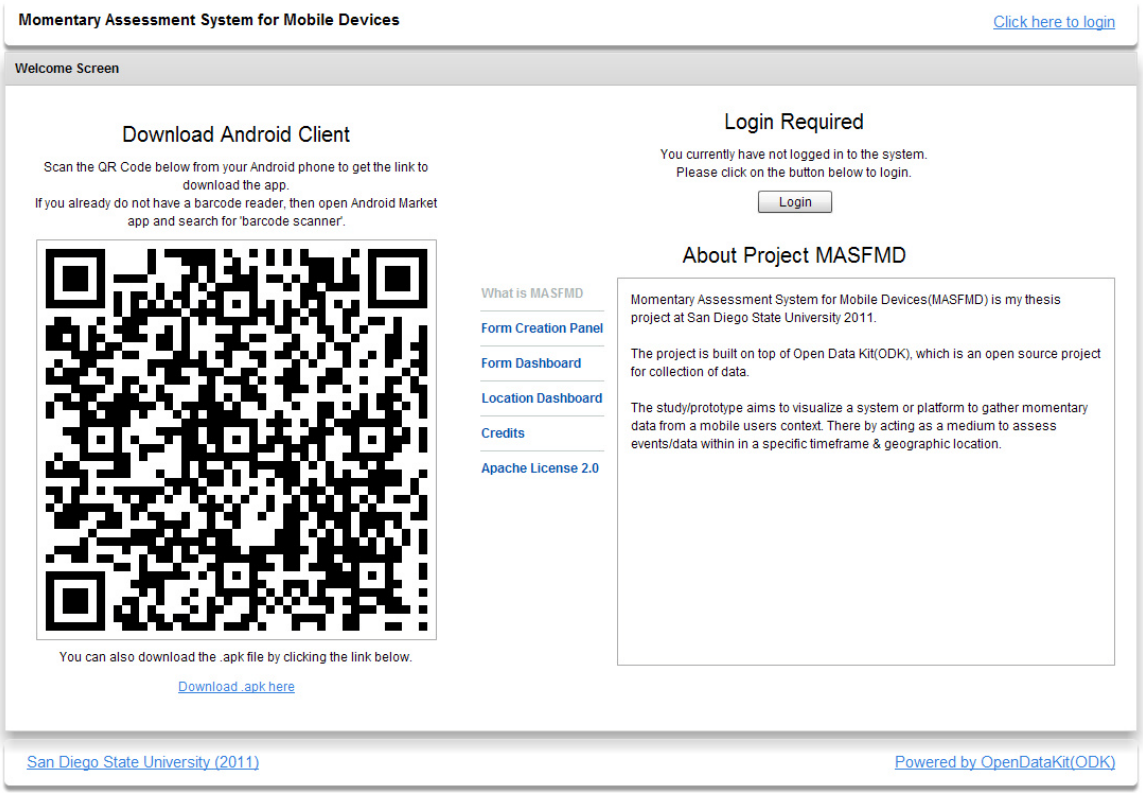

Momentary Assessment System for Mobile Devices
Click here to login
Welcome Screen
Download Android Client
Scan the QR Code below from your Android phone to get the link to download the app.
If you already do not have a barcode reader, then open Android Market app and search for 'barcode scanner'.
You can also download the .apk file by clicking the link below.
Download .apk here
Login Required
You currently have not logged in to the system.
Please click on the button below to login.
Login
About Project MASFMD
What is MASFMD
Form Creation Panel
Form Dashboard
Location Dashboard
Credits
Apache License 2.0
Momentary Assessment System for Mobile Devices(MASFMD) is my thesis project at San Diego State University 2011.
The project is built on top of Open Data Kit(ODK), which is an open source project for collection of data.
The study/prototype aims to visualize a system or platform to gather momentary data from a mobile users context. There by acting as a medium to assess events/data within in a specific timeframe & geographic location.
San Diego State University (2011)
Powered by OpenDataKit(ODK)

Figure B.1. Homepage of my master's thesis project.